

LARAVEL

Creating Your First Laravel App and Adding Authentication



Holly Lloyd
R&D Content Engineer

**Prosper Otemuyiwa**

Former Auth0 Employee

Last Updated On: January 09, 2020

👉 **Laravel 6.0 is out! Check out [Build a Laravel 6 CRUD App with Authentication](#) to learn what new things Laravel can do for you.**

🔔 **This blog post covers Laravel 5.8.**

TL;DR: Laravel is a great PHP framework. Currently, it is the most starred PHP project on [Github](#) and a lot of companies and people all over the world use it to build amazing applications. In this tutorial, I'll show you how easy it is to build a web application with Laravel and add authentication to it without breaking a sweat. Check out the [repo](#) to get the code.

Laravel is a free, open-source PHP framework designed for building web applications with an expressive and elegant syntax. **Laravel** has a high level of abstraction which shields the common developer from complex inner workings. **Laravel** saves you time and effort because it ships with a lot of features out of the box. These amazing features include:

- Database Migrations
- Eloquent ORM
- Authorization and Policies
- Scheduler
- Queuing

Laravel makes good use of already written and well-tested components from the PHP community. It is one of the few frameworks that comes with development environments such as [Homestead](#) and [Valet](#). The [documentation](#) is very detailed and there is a large community based around

Laravel. Some of the notable communities are laracasts.com, larajobs.com, laravel-news.com, laravelpodcast.com and larachat.co.

"Laravel is one of the few frameworks that actually with development environments such as

Homestead"



We'll be building a simple character listing app with **Laravel 5.8**. Our app will simply list **10 Game of Thrones characters** and their real names. Once we add authentication to the app, all logged-in users will have the privilege of knowing these celebrity characters personally.

Let's Get Started

Laravel utilizes [Composer](#) to manage its dependencies. So, before using Laravel, make sure you have Composer installed on your machine. We can install Laravel by issuing the Composer `create-project` command in your terminal like so: `composer create-project --prefer-dist laravel/laravel GOT` or using the `laravel` installer.

It's faster to spin up a new app using the `laravel` command like so: `laravel new GOT`. Check out the [Laravel docs](#) to learn how to set up the Laravel installer.

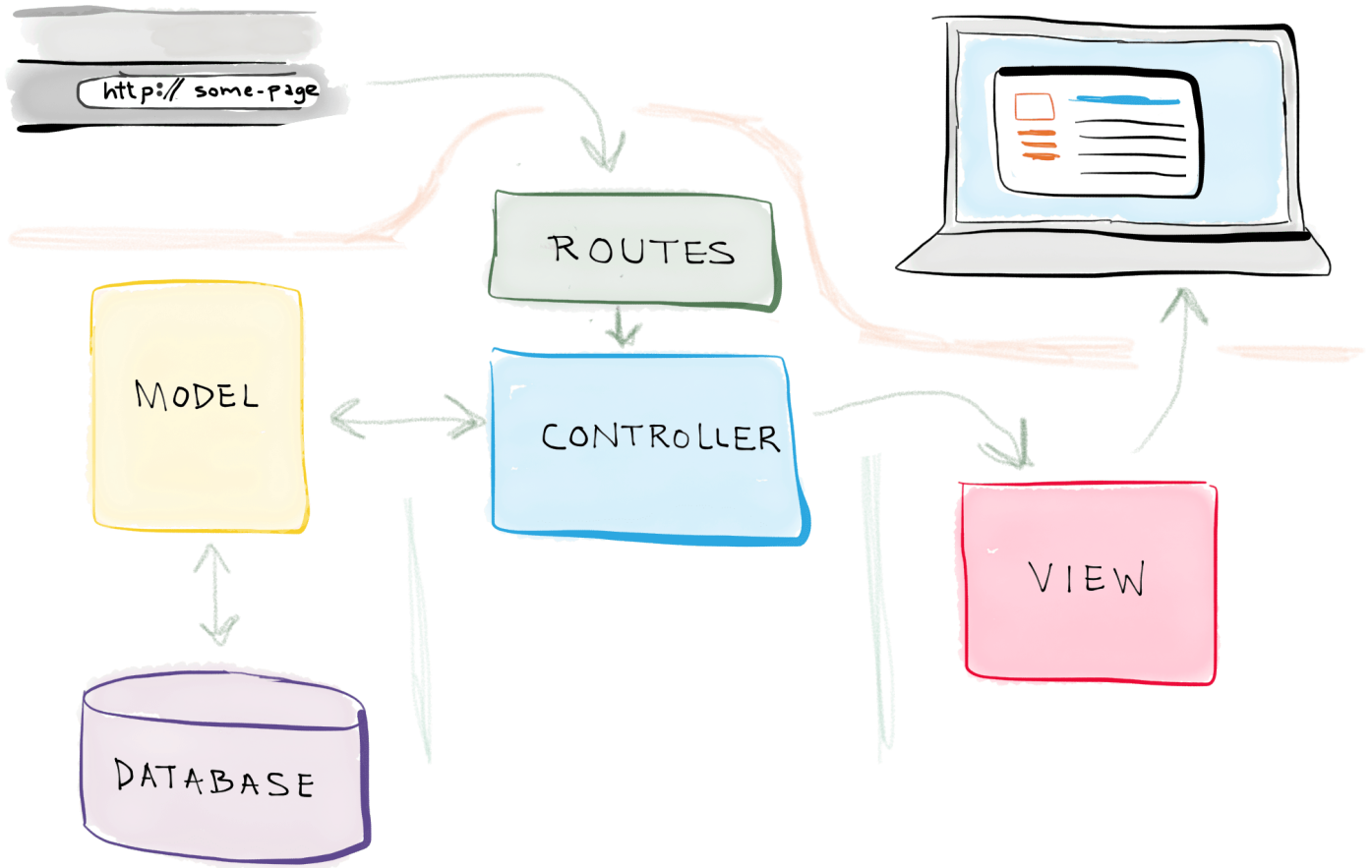
If you used the `laravel` installer command to create a new app, then you have to run `composer install` immediately after the previous command to install all the dependencies.

Now run the following in your terminal to launch your application:

```
php artisan serve
```

Explore Directory Structure

Laravel applications follow the **Model-View-Controller** design pattern.



(Source: [Self Taught Coders](#))

In a nutshell,

- **Models** query your database and return the necessary data.
- **Views** are pages that render data
- **Controllers** handle user requests, retrieve data from the models, and pass them to the views.

Read more about [MVC](#) here.

The app directory is the **meat** of your Laravel application. It houses the following directories:

- `Console` - Contains all your Artisan commands
- `Http` - Contains all your controllers, middleware, requests, and routes file

- `Providers` - Contains all your application service providers. You can read more about [Service Providers](#) here
- `Events` - Contains all your event classes
- `Exceptions` - Contains your application exception handler and custom exception classes
- `Jobs` - Contains all the jobs queued by your application
- `Listeners` - Contains all the handler classes for your events
- `Policies` - Contains the authorization policy classes for your application. Policies are used to determine if a user can perform a given action against a resource.

The other directories namely:

- `bootstrap` contains your framework autoloading files and generated cache files
- `config` contains your app's configuration files
- `database` contains your database migrations and seeds
- `public` contains your assets (images, JavaScript, css, etc.)
- `resources` contains your views and localization files
- `storage` contains all your compiled Blade templates, file caches, and logs
- `tests` contains all your tests
- `vendor` contains your app dependencies

Setting Up The Controller

Open up your terminal and in the project root directory, run the command below to create a `ListController`.

```
php artisan make:controller ListController
```

Open up `app/Http/Controllers/ListController.php` and configure it like so:

```
<?php

namespace App\Http\Controllers;
```

```
class ListController extends Controller
{
    public function show()
    {
        $characters = [
            'Daenerys Targaryen' => 'Emilia Clarke',
            'Jon Snow'           => 'Kit Harington',
            'Arya Stark'         => 'Maisie Williams',
            'Melisandre'         => 'Carice van Houten',
            'Khal Drogo'         => 'Jason Momoa',
            'Tyrion Lannister'   => 'Peter Dinklage',
            'Ramsay Bolton'     => 'Iwan Rheon',
            'Petyr Baelish'      => 'Aidan Gillen',
            'Brienne of Tarth'   => 'Gwendoline Christie',
            'Lord Varys'         => 'Conleth Hill'
        ];

        return view('welcome')->withCharacters($characters);
    }
}
```

`view('welcome')->withCharacters($characters)` indicates that we are passing the `$characters` array to a view called `welcome.blade.php`. We'll create that view later in this post.

Setting Up The Model

Laravel models are stored by default in the root of the `app` directory. The `User` model ships with the Laravel framework. Only the `User` model is needed in this application so we won't create any additional models. However, if you want to create more models, you can simply run the command below like so:

```
php artisan make:model <modelName>
```

where `<modelName>` represents the name of the model you want to create.

Setting Up The Routes

Open up `routes/web.php` and configure it like so:

```
/*
|-----
| Application Routes
|-----
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

Route::get('/', 'ListController@show');
```

Once a request hits the `/` route, it invokes the `show` method of the `ListController` and renders the returned value in the `welcome` view. We'll configure the `welcome` view later in this post.

Setting Up Authentication

We're going to be using Auth0 for authentication. Setting up the built-in authentication with Laravel is pretty **straightforward, but limited**. With Auth0, you'll have access to an easy-to-use dashboard, the ability to integrate social identity providers, two-factor authentication, passwordless login, and more. And luckily, it's just as easy to integrate with your Laravel application! Let's check it out.

If you don't already have an account, go ahead and [sign up for a free Auth0 account here](https://auth0.com/blog/creating-your-first-laravel-app-and-adding-authentication/).

After you've signed up, head to the dashboard and click on "Applications". Click on "Create Application" and name the application "Laravel App" or anything you'd like. Then click on "Regular Web Application" and press create.

Next, you need to add the valid callback URLs and logout URLs in the dashboard.

Click on "Applications" and select the application you just created (or the default one). Next, click on "Settings".

Update these values as follows:

- Allowed callback URLs: `http://localhost:8000/auth0/callback`
- Logout URLs: `http://localhost:8000`

Next, go back to your terminal and install the Auth0 PHP plugin and Auth0 Laravel plugin:

```
composer require auth0/login:"~5.0"
```

Now open the `config/app.php` file and add the Auth0 login service provider to the list of providers:

```
// ...  
'providers' => [  
    // ...  
    Auth0\Login\LoginServiceProvider::class,  
];
```

Then scroll down to the `aliases` array and add the Auth0 facade:

```
// ...  
'aliases' => [  
    // ...
```



```
'Auth0' => Auth0\Login\Facade\Auth0::class,  
];
```

Next, open `app/Providers/AppServiceProvider.php` and add the following under `register()`:

```
// ...  
class AppServiceProvider extends ServiceProvider  
{  
    // ...  
    public function register()  
    {  
        $this->app->bind(  
            \Auth0\Login\Contract\Auth0UserRepository::class,  
            \Auth0\Login\Repository\Auth0UserRepository::class  
        );  
    }  
}
```

You need to publish the plugin configuration. In your terminal, run:

```
php artisan vendor:publish
```

When it asks which file you'd like to publish, select `Auth0\Login>LoginServiceProvider`.

This will create the `config/laravel-auth0.php` file. Open this up now and you'll see it's using some Auth0 variables. These are sensitive, so you need to add them to the `.env` file.

Open up `.env` and add:

```
AUTH0_DOMAIN=your-auth0-domain.auth0.com  
AUTH0_CLIENT_ID=your-client-id
```

```
AUTH0_CLIENT_SECRET=your-client-secret
```

To fill in these values, head back to [your Auth0 dashboard](#), select your application, and click on "Settings".

While you have `.env` open, double-check that `APP_URL` matches your dev URL exactly, **including the port**. It should say `APP_URL=http://localhost:8000`. If not, update that now.

The last configuration step is to switch out the Laravel user driver to use Auth0.

Open up `config/auth.php`, scroll down to `providers`, and paste in:

```
'providers' => [  
    'users' => [  
        'driver' => 'auth0',  
    ],  
],
```

Now that you have the Auth0 plugin configured, you just need to integrate it into the application.

Integrating Auth0

Open up `routes/web.php` and add these authentication routes:

```
Route::get( '/auth0/callback', '\Auth0\Login\Auth0Controller@callback' )->name('auth0.callback');  
Route::get( '/login', 'Auth\Auth0IndexController@login' )->name( 'login' );  
Route::get( '/logout', 'Auth\Auth0IndexController@logout' )->name( 'logout' );
```

These routes will handle the login, logout, and redirect to Auth0 during login. Now you need to create that `Auth0IndexController`.

In your terminal, run:

```
php artisan make:controller Auth/Auth0IndexController
```

Open up `app/Http/Controllers/Auth/Auth0IndexController.php` and replace it with:

```
<?php

namespace App\Http\Controllers\Auth;

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

class Auth0IndexController extends Controller
{
    /**
     * Redirect to the Auth0 hosted login page
     *
     * @return mixed
     */
    public function login()
    {
        $authorize_params = [
            'scope' => 'openid profile email',
        ];

        return \App::make('auth0')->login(null, null, $authorize_params);
    }

    /**
     * Log out of Auth0
     *
     * @return mixed
     */
    public function logout()
```

```
{
    \Auth::logout();
    $logoutUrl = sprintf(
        'https://%s/v2/logout?client_id=%s&returnTo=%s',
        env('AUTH0_DOMAIN'),
        env('AUTH0_CLIENT_ID'),
        env('APP_URL'));
    return \Redirect::intended($logoutUrl);
}
```

The scopes being requested in the `login()` function are: `openid`, `profile`, and `email`.

There is also a `logout()` function that will clear all session data to log the user out.

Now it's time to wire up the buttons so that users can sign in and out.

Open up your `welcome.blade.php` and configure it like so:

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Laravel</title>
    </head>
    <body>
        <div class="flex-center position-ref full-height">
            @if (Route::has('login'))
                <div class="top-right links">
                    @if(Auth::user())
                        <a href="{{ url('/home') }}">Home</a>
                        <a href="{{ url('/logout') }}">Logout</a>
                    @endif
                </div>
            @endif
        </div>
    </body>
</html>
```

```

        @else
            <a href="{{ route('login') }}">Login</a>
        @endif
    </div>
@endif

<div class="container">
    <div class="row">
        <div class="col-md-10 col-md-offset-1">
            <div class="panel panel-success">
                <div class="panel-heading">List of Game of Thrones Characters</div>
                @if(Auth::user())
                    <!-- Table -->
                    <table class="table">
                        <tr>
                            <th>Character</th>
                            <th>Real Name</th>
                        </tr>
                        @foreach($characters as $key => $value)
                            <tr>
                                <td>{{ $key }}</td><td>{{ $value }}</td>
                            </tr>
                        @endforeach
                    </table>
                @endif
            </div>
            @if(Auth::guest())
                <a href="/login" class="btn btn-info"> You need to login to see th
            @endif
        </div>
    </div>
</div>
</body>
</html>

```

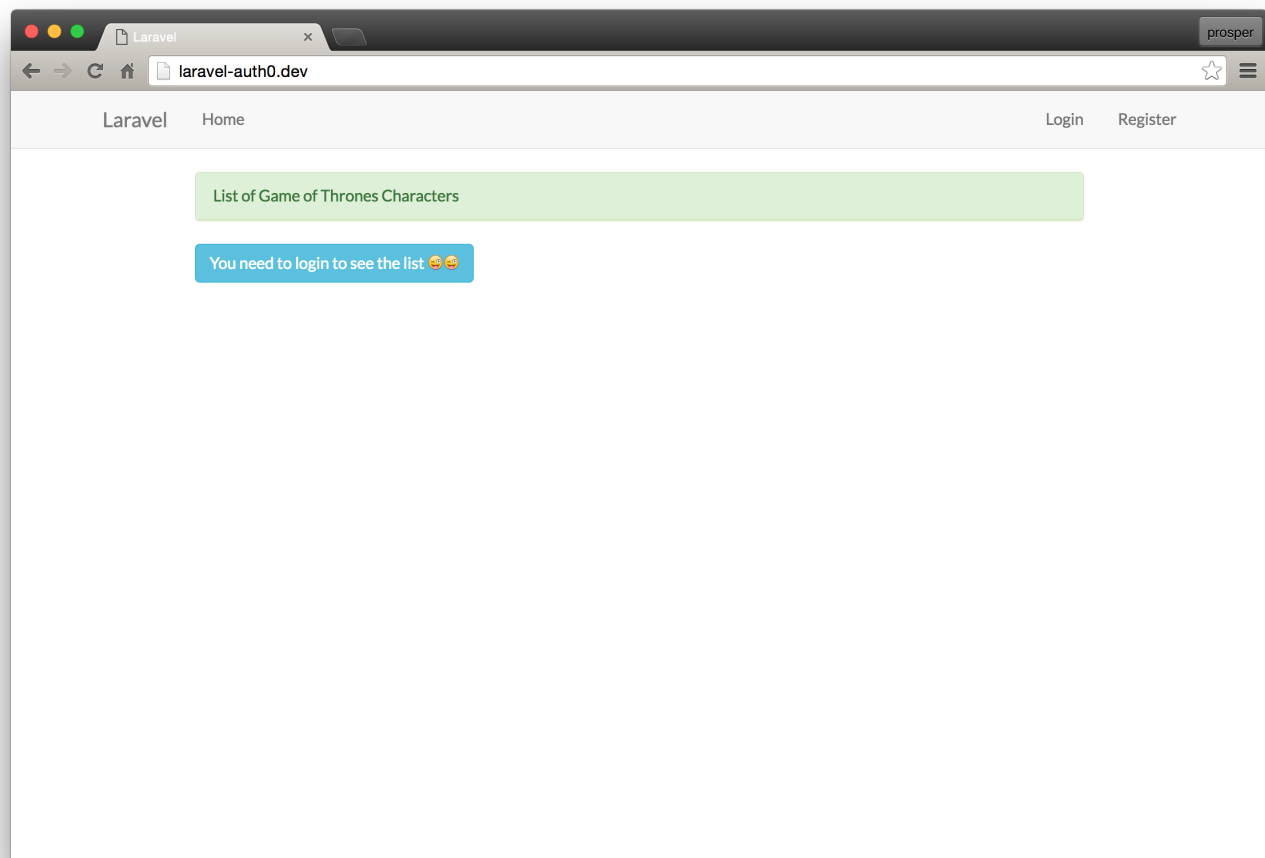
Here, we are looping through the `$characters` array data passed from the `ListController` for appropriate rendering in the `welcome` view.

`Auth::user()` — You can check if a user is authenticated or not via this method from the `Auth` Facade. It returns true if a user is logged-in and false if a user is not. Check [here](#) for more about how Facades work in Laravel.

`Auth::guest()` — This does the opposite of `Auth::user()`. It returns true if a user is not logged in and false if a user is logged in. Check [here](#) to see all the methods you can call on the `Auth` Facade.

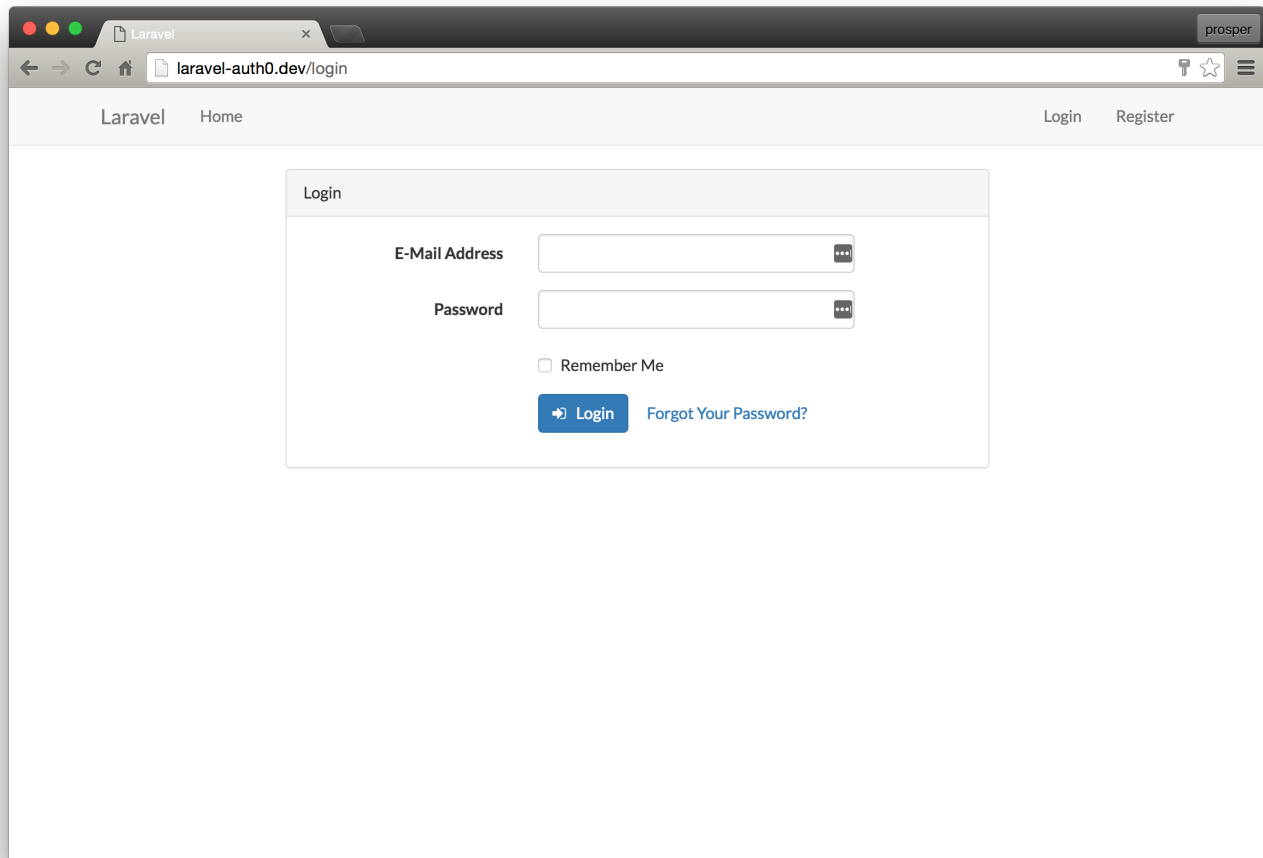
Now that we have all the routes and views setup, your application should look like this:

Landing Page



If you were to implement authentication on your own, you might have a login and register page like this:

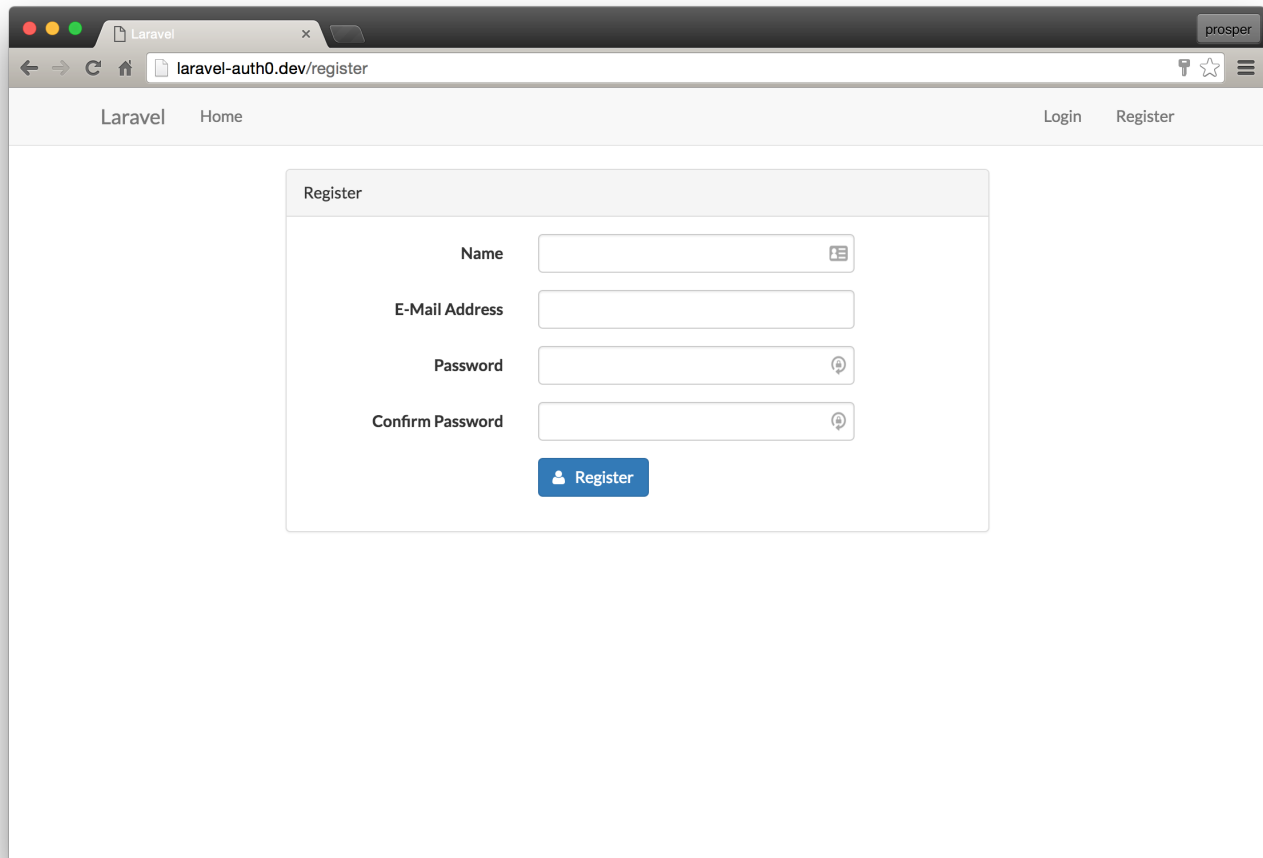
Login Page



The screenshot shows a web browser window with the address bar displaying 'laravel-auth0.dev/login'. The page has a header with 'Laravel' and 'Home' on the left, and 'Login' and 'Register' on the right. The main content area features a 'Login' form with the following elements:

- A title 'Login' at the top of the form box.
- An 'E-Mail Address' label next to a text input field.
- A 'Password' label next to a password input field with a toggle icon.
- A checkbox labeled 'Remember Me'.
- A blue 'Login' button with a right-pointing arrow.
- A link labeled 'Forgot Your Password?'.

Register Page



Because we're using Auth0 for authentication, our application will redirect users to the Auth0 login page, so you don't have to create these on your own!

Using the Auth Middleware

Middlewares provide a convenient mechanism for filtering HTTP requests entering your application. For example, **Laravel** includes a middleware that verifies the user of your application is authenticated. If the user is not authenticated, the middleware will redirect the user to the login screen. However, if the user is authenticated, the middleware will allow the request to proceed further. The `app/Http/Middleware` directory contains several middleware.

Let's check out how the `auth` middleware works.

Add a new route to your `routes/web.php` file like so:


```
Route::get('/got', [  
    'middleware' => ['auth'],  
    'uses' => function () {  
        echo "You are allowed to view this page!";  
    }  
]);
```

If you're still logged in, head to <http://localhost:8000/got> and you should get the message "You are allowed to view this page!". Now, log out, then try to access that route and you will be redirected back to Auth0 to sign in.

The Laravel `auth` middleware intercepted the request, checked if the user was logged-in, discovered that the user was not logged-in, then redirected the user back to the `login` route, which sent them to Auth0.

Wrapping Up

Well done! You have just built your first app with Laravel. Laravel is an awesome framework to work with. It focuses on simplicity, clarity and getting work done. As we saw in this tutorial, you can easily activate the built-in authentication to your Laravel applications. If you find yourself needing more, you can also integrate Auth0 just as easily.

Please, let me know if you have any questions or observations in the comment section. 😊

AUTH0 DOCS [🔗](#)

Implement Authentication in Minutes

Auth0 Marketplace

Discover and enable the integrations
you need to solve identity

Explore Auth0 Marketplac



More like this



RUBY

Ruby Authentication: Secure Your Rack Application with JWT

GOLANG

Authentication in Golang with JWTs



VUE

Build an App with Vue.js: From Authentication to Calling an API

Follow the conversation



48 Comments Auth0 Blog Disqus' Privacy Policy

Login ▾

Recommend 11 Tweet Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS

**Renato Leite** • 2 years ago

is this still working?

Is there any other tutorial with the new versions?

2 ^ | ▾ • Reply • Share ›

**Holly Lloyd** Mod ➔ Renato Leite • a year ago

Hi Renato! This tutorial has been updated to Laravel 5.8. If you'd like an even more in depth and newer one, we just published this one that's compatible with Laravel 6.0:

<https://auth0.com/blog/buil...>

^ | ▾ • Reply • Share ›

**Dan Arias** Mod ➔ Renato Leite • 2 years ago

Howdy, Renato Leite !

Our newest Laravel content includes GraphQL:

<https://auth0.com/blog/deve...>

We have it in our pipeline to publish an updated version of this post, I'd love if you could please share with me what would you like to see in such a post?

Thank you for reading and your feedback!

^ | ▾ • Reply • Share ›

**Photolancer Zone** • 4 years ago

Nice article, thank you!

An easy way to try out Laravel in action is to use Laravel 5 Boilerplate / Starter Kit -

<https://github.com/Labs64/l...>

This is also offering Docker container, with this you don't need a local PHP (composer, node.js, etc.) environment and can start to evaluate Laravel right away.

2 ^ | v • Reply • Share ›



Eze Sunday • 4 years ago • edited

Well, this helped me alot to get started with laravel.

Thank you Prosper.

Achieved much in a short time.

3 ^ | v 1 • Reply • Share ›



Prosper → Eze Sunday • 4 years ago

I'm glad this helped!

1 ^ | v • Reply • Share ›



Akash Sharma • a year ago

yeah !! its is a informational Post. We can add authentication with middleware also in laravel and Multiple auth in laravel.

2 ^ | v 1 • Reply • Share ›



Stop Medicaid Theft • 3 years ago • edited

In the "Setting Up The Controller" section of this article it says, "Open up your terminal and run the command below to create a ListController".

Type "php artisan make:controller ListController".

What directory should I be in when I type "php artisan make:controller ListController"?

1 ^ | v • Reply • Share ›



Prosper → Stop Medicaid Theft • 3 years ago

@Stop Medicaid Theft The root directory

1 ^ | v • Reply • Share ›



Alex Kramer • 3 years ago • edited

Hi, thanks for this tutorial it helped me get started alot quicker.

However the list appears to remain blank (the view shows) after some scanning through your work I figured it was because of the line in the foreach loop in your view:

Shouldn't this line:

```
<td></td><td></td>
```

Be:

```
<td>?php echo $key ?</td><td>?php echo $value ?</td>
```

That is what's working for me, but if it's a wrong approach I would still like to know

EDIT: I changed the start tag for PHP to just the ? because otherwise it would disappear!

That is probably what happened to your code too ?

1 ^ | v • Reply • Share ›



Prosper → Alex Kramer • 3 years ago • edited

Hello **@Alex Kramer** please check the source code here <https://github.com/auth0-bl...>

The blog engine didn't render the code in that section. I'll find a way to escape it now and update the post so that it can be rendered properly. Sorry for the inconvenience.

2 ^ | v • Reply • Share ›



Marcos Saldivar • 4 years ago

Hello, Prosper. What if I use an LDAP authentication provider? How would it be?
Thank you for your attention, Marcos.

1 ^ | v • Reply • Share ›



James Moore • 4 years ago • edited

Great Tutorial....

Note: It would be suggested to update the View.... so; instead of the Looping Construct looking like this:

```
@foreach($characters as $key => $value)
<tr>
<td></td><td></td>
</tr>
@endforeach
```

The Looping Construct looks rather like this:

```
@if (isset($characters))
@foreach($characters as $key => $value)
<tr>
<td>{{$key}}</td>
<td>{{$value}}</td>
</tr>
@endif
```

Otherwise, a Nice tutorial though.....

1 ^ | v 1 • Reply • Share ›



foysal • 4 years ago

I followed your tutorial. I used Laravel 5.3.10. You did not mention here the logout process. How user logout from the application. I faced the issue and made a post (<http://stackoverflow.com/qu....> It is better if you help me. Thanks

1 ^ | v 1 • Reply • Share ›



Prosper → foysal • 4 years ago

Are you still having this issue? Replied on the stackoverflow thread

^ | v • Reply • Share ›



alexander_voronkov • 3 months ago

It doesn't work with Laravel 8.0 at all

^ | v • Reply • Share ›



mahedy • a year ago

Great article thumbs up

^ | v • Reply • Share ›



Sam Watson • a year ago

Yeah you have share a great information about Laravel app creation and authentication I will try to implement this. on Laravel development keep posting....

^ | v • Reply • Share ›



Holly Lloyd Mod ➔ **Sam Watson** • a year ago

Hey Sam! We just published another Laravel authentication tutorial for the new 6.0 release. You can check that one out if you'd like something a little more up to date!
<https://auth0.com/blog/buil...>

^ | v • Reply • Share ›



Maximiliano Chiale • a year ago

why the login method is different with <https://auth0.com/docs/quic...> this on the guide??

^ | v • Reply • Share ›



Holly Lloyd Mod ➔ **Maximiliano Chiale** • a year ago

Hey Maximiliano! This tutorial is actually just showing how to use Laravel's built-in authentication. If you'd like to use Auth0, we just published a new tutorial on how to integrate Auth0 with Laravel 6.0!

<https://auth0.com/blog/buil...>

^ | v • Reply • Share ›



Ravikumar Phad • 2 years ago • edited

Very Nice Tutorial Prosper....Thank you very Much.

^ | v • Reply • Share ›



Nay Oo Kyaw • 3 years ago

Very Great topic in Laravel

^ | v • Reply • Share ›



Izabel • 3 years ago

Hi Prosper!The tutorial is great, appreciate for your time!

I am new to Laravel and I faced the following problem:

I made the installation on web hosting and execute commands using ssh. Everything went fine until executing the following:

```
php artisan migrate
```

I made modifications to database.config file and .env file which are currently pointing to my sql server on a different server and it throws the following error:

```
[Illuminate\Database\QueryException]  
SQLSTATE[HY000] [2002] Connection refused (SQL: select * from information_s  
chema.tables where table_schema = Carrot and table_name = migrations)'
```

Do I need to make any modifications on that database? I read many articles, however I was unable to find the answer.

Can you please assist with that problem?

^ | v • Reply • Share ›



Gia Thiện • 3 years ago

Great!

^ | v • Reply • Share ›



Kunal • 3 years ago

I just followed your tutorial to create an application in laravel 5.4 using composer. I followed your steps until the migrations properly. But I am getting a notfoundexception error as i click on login page or the register page. Please provide an explanation for the same.



^ | v • Reply • Share ›



Kamil Zaworski • 4 years ago

.....



Hi I have an error:

ErrorException in AliasLoader.php line 66:
Class 'Auth0\Login\Facade\Auth0' not found

What could be the reason?

^ | v • Reply • Share ›



kylesean • 4 years ago

well,thanks for your tutorial.By the way,could you tell what's the tool used to paint that mvc picture ?

^ | v • Reply • Share ›



adobot → kylesean • 4 years ago

That image comes from SelfThoughtCoders. Might want to check out their site and see if they share that info :)

^ | v • Reply • Share ›



el_charlie • 4 years ago

Hi,

I'm new in this Laravel thing but I can't follow the tutorial properly. I don't have the app/Http/routes.php file.

What can I do?

^ | v • Reply • Share ›



adobot → el_charlie • 4 years ago

Are you starting up your application with Composer? If so the routes.php file should be created automatically. Let me know if that works.

^ | v • Reply • Share ›



el_charlie → adobot • 4 years ago

Either with composer or when using "laravel new <projectname>" I don't get the routes.php. I do have a routes folder and some phps in there.

^ | v • Reply • Share ›



Prosper → el_charlie • 4 years ago

What version of laravel are you using?

If you are using 5.3 or 5.4, then there is no routes.php file, rather it will give you a routes folder. Inside the routes directory, you'll see an api.php, console.php and web.php file...that's where your routes are defined.

Note: This tutorial was developed with Laravel 5.2

1 ^ | v • Reply • Share ›



Eric Rosenberg → Prosper • 4 years ago

Can you share what to do in the "Setting Up The Routes" section on the newer versions? I opened all three of the new routes files but don't see a section for application routes.

^ | v • Reply • Share ›



Eric Rosenberg → Eric Rosenberg • 4 years ago

I figured this one out. I added it to the web.php file and everything is working.

^ | v • Reply • Share ›



Avatar

This comment was deleted.



Binod Kumar → Guest • 3 years ago

never seen a easy tutorial of laravel like this god bless ya...

^ | v • Reply • Share ›



syam • 4 years ago

sorry.. can you help me?? i need this project, can i download this master file??

^ | v • Reply • Share ›



Kim Maida → syam • 4 years ago

Yes, absolutely! The source code for this tutorial is available here:
<https://github.com/auth0/la...> Cheers!

^ | v • Reply • Share ›



Sacchidanand Tiwari → Kim Maida • 4 years ago

But Sql file is not available.

^ | v • Reply • Share ›



adobot → Sacchidanand Tiwari • 4 years ago

Hi There, you will find the database migration files here:
<https://github.com/auth0/la...>

Thanks!

^ | v • Reply • Share ›



Adam Tesluk • 4 years ago • edited

I followed the tutorial. Do you know what causes this error?
Undefined variable: line 15 in welcome.blade.php

^ | v • Reply • Share ›



Adharul M → Adam Tesluk • 4 years ago

Perhaps you forgot to write in routes/web.php :

```
Route::get('/', 'ListController@show');
```

^ | v • Reply • Share ›



Daniel Soulz • 4 years ago

really thanks :) please write more ,cant wait to read another great posts. Thanks again!

^ | v • Reply • Share ›



Awadhesh Kumar • 4 years ago

Hi,

I followed this tutorial. I am using Laravel 5.3, I have facing a problem to connecting database, can u help me please

^ | v • Reply • Share ›



Jay Mayu → Awadhesh Kumar • 4 years ago

open the .env file and update your db connection details.

48 ^ | v • Reply • Share ›



Francis Rodrigues • 4 years ago

There are a problem during login process. The user is redirected by /login/callback, but is true it could be redirected by "/auth0/callback".

^ | v • Reply • Share ›



Prosper → Francis Rodrigues • 4 years ago

Hi Francis, what exactly seems to be the challenge here? Could you please

Secure access for everyone. But not just anyone.

TRY AUTH0 FOR FREE

TALK TO SALES

BLOG

Developers

Identity & Security

COMPANY

About Us

Customers

Business

Culture

Engineering

Announcements

Security

Careers

Partners

Press

PRODUCT

Single Sign-On

Password Detection

Guardian

M2M

Universal Login

Passwordless

MORE

Auth0.com

Ambassador Program

Guest Author Program

Auth0 Community

Resources



© 2013-2021 Auth0 Inc. All Rights Reserved.