# [Multiple Authentication in Laravel 5.4](https://www.youtube.com/playlist?list=PLwAKR305CRO9S6KVHMJYqZpjPzGPWuQ7Q)

Create a new project:

>laravel new multiauth

Creating basic authentication

>php artisan make:auth

Authentication scaffolding generated successfully.

It will automatically creates the HomeController with the middleware, views like auth, layouts,home.blade.php and routes -

..

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

Creating migration for admins table:-

>php artisan make:migration create\_admins\_table

Created Migration: 2019\_05\_09\_202557\_create\_admins\_table

Creating fields for admins table:-

...

public function up()

{

Schema::create('admins', function (Blueprint $table) {

$table->bigIncrements('id');

$table->string('name');

$table->string('email')->unique();

$table->string('job\_title');

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

Create database “multiauth”

Set varibales in env file

DB\_DATABASE=multiauth

DB\_USERNAME=root

DB\_PASSWORD=

>php artisan migrate

Migration table created successfully.

Migrating: 2014\_10\_12\_000000\_create\_users\_table

Migrating: 2014\_10\_12\_100000\_create\_password\_resets\_table

Migrating: 2019\_05\_09\_202557\_create\_admins\_table

Making Model:-

We copy the default User model - User.php and make our Admin model - Admin.php

Now we have two models one for users and one for admins

Is uses Notifiable for using the notificaiton functionality in Application.

It determines based on the user who to send notification to and how to send it .

use Illuminate\Foundation\Auth\User as Authenticatable;

This is our base model for authenticable users, it itself extends the Model class.

Adding job\_title in fillable array, Everyting in fillable array is mass assignable.

protected $fillable = [

'name', 'email', 'password',**'job\_title'**,

];

Hidden from the collection of the array, its put in $hidden array, so that it doesn’t get thrown in with all the other stuff

protected $hidden = [

'password', 'remember\_token',

];

Setting up Guard:-

Config>auth.php

Laravel have multiple ways to log people in, so we call each of these ways as guards, default is web guard - http authentication, other type is API guard.

'defaults' => [

'guard' => 'web',

'passwords' => 'users',

],

We can add as many guards as we want to en-tying it to the model we have created ie, Admin model

E.g-

Auth::check($credentials) - check a user is authentic or not

Auth::attempt($credentials) - use to log people in

Auth::once($credentials) - log in for one HTTP request

…etc

They gonna use default guard- web, if we don’t specify a guard.

Using the admin(we gonna create it) guard instead of default:

Auth::guard(‘admin’)->check($credentials)

Two Ways to track a guard:-

'guards' => [

'web' => [

'driver' => 'session', **->using session**

'provider' => 'users',

],

'api' => [

'driver' => 'token', **->using token**

'provider' => 'users',

'hash' => false,

],

],

Provider tells how to talk back-forth with the database

Two drivers to talk back-forth with database

'providers' => [

'users' => [

'driver' => 'eloquent', **->elequent ORM**

'model' => App\User::class,

],

// 'users' => [

// 'driver' => 'database', -**>database driver uses query builder**

// 'table' => 'users',

// ],

],

Configure a new providers for the admins so that it knows how to talk to the admins database.

'admins' => [

'driver' => 'eloquent',

'model' => App\Admin::class,

],

Configure a new guard for the admin

'admin' => [

'driver' => 'session',

'provider' => 'admins',

],

Setting a guard for admin API

'admin-api' => [

'driver' => 'token',

'provider' => 'admins',

'hash' => false,

],

Configuring password reset for admins

'admins' => [

'provider' => 'admins',

'table' => 'password\_resets',

'expire' => 15,

],

Configuring Admin model so that it knows which guard to work with

Configuring admin guard in Admin model:-

…

protected $guard = 'admin';

...

Creating AdminController :-

Duplicating the Homecontroller and changing its parameters.

Creating view for AdminController@index method admin.blade.php

Duplicating the home view home.blade.php to create admin.blade.php

And changin content inside.

**AdminController.php**

Specifying guard in authentication middleware:-

public function \_\_construct()

{

$this->middleware('auth');

}

It is tied by authentication middleware by default, if we don’t specify a guard then it’s going to default guard. In our case its ‘web’ which uses ‘users’ provider.

So it will allow any login users to come in and visit /admin page and other admin pages.

So we need to define the guard we want to protect against in our middleware, in our case we want the admin guard.

public function \_\_construct()

{

$this->middleware('auth**:admin**');

}

In Laravel 5.4 we actually can natively support multiple User models (Sometimes called MultiAuth). This means we can have different users and manage these users independently, all using the Native Auth Facades, without any packages or plugins. An example of this would be Employees that manage a backend interface and then customers which have the ability to log in and manage accounts, but through the front end of the website. We want to use different tables to manage these types of users, and have different middleware and guards in place as well. This tutorial will show multiple authentication to you.

###### Creating seprate login form for Admins

The current login form will login us for customers/users and not login for admins.

The current login form can be used only for User model and not for Admin model.

The current login form is tied to User model.

\*We can use the same logout function for both of these,. User and Admin model/Users . Logout works universally, so it logs us out of all guards resets/clears the session and then sets up a brand new clean session and then redirects us home.

\*registering admins is kinda weird . Admins will not have the ability to register themself, a super user can create admins and they can login themself.

1. Creating a Controller to manage Admin login inside Auth Controller.

Creating an AdminLoginController

>php artisan make:controller Auth/AdminLoginController

Controller created successfully.

Creating showLoginForm() and login()

class AdminLoginController extends Controller

{

public function showLoginForm()

{

return view('auth.admin-login');

}

public function login(){

return true;

}

}

Creating Views

Views>auth>admin-login.blade.php

Duplicating the login.blade.php and making changes

Calling its header ADMIN Login and changing form submit route to admin.login.submit

Creating Routes

Route::get('admin/login', 'Auth/AdminLoginController@showLoginForm')->name('admin.login');

Route::post('admin/login', ['Auth/AdminLoginController@login')->name('admin.login.submit');](mailto:'Auth/AdminLoginController@login')->name('admin.login.submit');)

Grouping all admin/ routes together

Route::prefix('admin')->group(function(){

Route::get('/login', 'Auth/AdminLoginController@showLoginForm')->name('admin.login');

Route::post('/login', 'Auth/AdminLoginController@login')->name('admin.login.submit');

Route::get('/','AdminController@index')->name('admin.dashboard');

}

\*The default guard for the authentication is web. Web middleware is different then web guard.

Web guard protecting against the user.

Web middleware is all functions that we need for all the browser-based visit to our website, like session, cookies etc.

**Setting middleware in our AdminLoginController:**

This controller can be only accessed by people that are not logged in as admin

public function \_\_construct(){

$this->middleware('guest:admin');

}

**Setting up Login function of AdminLoginController**

…

use Auth;

…

public function login(Request $request){

//validate the from data

$this->validate($request, [

'email' => 'required|email',

'password' => 'required|min:6'

]);

//Attempt to log the user in

Auth::attempt($credentials, $remember);

\*Use use the default attempt()method. Its the same default method that we see in the LoginController. It returns true-false on validation.

$credentials are passed in an array format, it the email and password, its going to verify every element in the array matches the model Admin.

$remember is the optional second parameter,true-false value for the remembering the user across multiple sessions always-once.

Auth::guard('admin')->attempt($credentials, $remember);

If we run normal method on this Auth facade without specifying the guard, it will use the default web guard. In our case we need to use Admin guard.

**if (Auth::guard('admin')->attempt(['email'=>$request->email, 'password'=>$request->password], $request->remember)) {**

**//if successful, then redirect to their intended location**

**return redirect()->intended(route('admin.dashboard'));**

**}**

**\*attempt() function automatically handels hashingin and checking it against the hash that stored in the db, so we don’t need to manually hash it.**

**It will trim the additional spaces from the form fields also except password and password\_confirmation -> Middleware>TrimString.php.**

\*intended() - redirect the visitor to the page where they were going before they got pulled out by middleware and had to login.Otherwise we set a default in case someone is just using the login form.

intended(route('admin.dashboard'))

//if unsuccessful, then redirect back to the login with the form data

return redirect()->back()->withInput($request->only('email', 'remember'));

It return visitors back to the login form if the login unsuccessful, it return them back with the $request data we can modify it to redirect with only email and remember and not password.

Creating a new admin in DB to login using artisan

>php artisan tinker

Psy Shell v0.9.9 (PHP 7.3.2 — cli) by Justin Hileman

>>> $admin = new App\Admin

=> App\Admin {#2949}

>>> $admin->name = "GauravPawar"

=> "GauravPawar"

>>> $admin->email = "hello@gauravpawar.io"

=> "hello@gauravpawar.io"

>>> $admin->password = Hash::make('password')

=> "$2y$10$8rsBWcc4CmbmLAmU7RCXMu4jg9hQpnP16HUrXnU1oTZ5E0QnOYrTG"

>>> $admin->job\_title = "Software Developer"

=> "Software Developer"

>>> $admin->save()

=> true

>>>

<http://localhost:8000/admin/login>

-login as admin

Username - [gauravpawar@hello.io](mailto:gauravpawar@hello.io)

Password - password

\*If you used middleware(‘guest’) it will NOT let you access the Admin Login from WHILE logged in as a User.

\*At this point we can login in as both User and Admin at same time

<http://localhost:8000/admin> -- Admin

<http://localhost:8000/home> --User

If we logout in one it will logout us in other too. Logout works as global and it will reset all sessions.

Admin Model - admin guard

User Model - web guard

localhost:8000/home(Home page for user) --> [http://localhost:8000/login( redirect](http://localhost:8000/login(redirect) to user login page)

**Problem 1 :**

localhost:8000/admin(Home page for admin) --> <http://localhost:8000/login> (redirect to user login page when we are not login as admin instead of admin login page “http://localhost:8000/admin/login”)

If we are logged in and if we are going to a page protected by guest middleware(eg. Login form page), it will redirect us to the dashboard/homepage for that user becaue they are not allowed to access the page protected by guest middleware(only guest can assess that page)

**Problem2:**

If we are login as admin and tries to acces the page protected by admin-guest middleware(I.e. the login page)

localhost:8000/admin/login(login page for admin) --> <http://localhost:8000/login(login> page for user)

Why is this happening?

Because we are protected by admin-guest middleware it thinks that we are already logged in , it falls back on a redirect and then it redirect to users dashboard which is hard coded URL(/home), the proble is that we are not authenticated as user so /home the kicks us out and send us to the user loign page.

app>http>kernel.php

…

protected $routeMiddleware = [

**'auth' => \App\Http\Middleware\Authenticate::class,**

'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,

'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,

'can' => \Illuminate\Auth\Middleware\Authorize::class,

**'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,**

'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,

'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,

];

…

Soluton1:

-------------------------------------------------Old Laravel 5.4-------------------------------------------------------------------------

Changes made in app\Exception\Handler.php

\*No such function present in my version of Laravel

protected function unauthenticated($request, AuthenticationException $exception)

{

If($request->expectsJson()){

return response()->json([‘error’ => ‘Unauthenticated.’], 401);

}

**$guard = array\_get($exception->guards(), 0);**

**switch($guard){**

**case ‘admin’:**

**$login = ‘admin.login’;**

**break;**

**default:**

**$login = ‘login’;**

**break;**

**}**

**//**return redirect()->guest(route(‘login’));

return redirect()->guest(route(**$login**));

}

-----------------------------------------end Old-------------------------------------------

**\App\Http\Middleware\Authenticate**

**This worked for me**

**REF: https://github.com/laravel/framework/issues/26292**

…

use Illuminate\Support\Facades\Route;

…

protected function redirectTo($request)

{

if (! $request->expectsJson()) {

**if (Route::is('admin.\*')) {**

**return route('admin.login');**

**}**

return route('login');

}

}

Soluton 2:

\App\Http\Middleware\RedirectIfAuthenticated

public function handle($request, Closure $next, $guard = null)

{

**switch ($guard) {**

**case 'admin':**

**if (Auth::guard($guard)->check()) {**

**return redirect()->route('admin.dashboard');**

**}**

**break;**

**default:**

if (Auth::guard($guard)->check()) {

return redirect('/home');

}

**break;**

}

return $next($request);

}

# Configuring Multi Auth Middleware

We have our logins working as intended and are able to log in and out as our different users. We have a Users model and Admins model, tracking different types of users independently. Now we simply need to fix a few weird occurrences.

The first problem we have is that if we ever try to go to our Admin center when we are not logged in, it redirects us to the Users Login Form. This is not right, we would expect the app to redirect us to the Admin Login Form so that we can log in as an Admin. We will fix this part first. This is can be handled in our exceptions handler.

The second problem is that when our "Guest" middleware notices a logged in user, it always redirects us to the /home location. This is ok when we are accessing a users guest path, but if we are trying to access an admin guest path, this is very unexpected. The more natural concept would be to redirect us to the admin dashboard.

We can edit this functionality in the RedirectIfAuthenticated Middleware. Just like before, we use our guard and test if it is either admin or web. Then redirect to the correct url for that guard's dashboard.

Logging out multiple users

creating a user interface module which shows which user is logged in and wihch is logged out:

We will create this module as component in laravel.

Components are reusable blade pages almost like a partial but it allows us to input them into sections

Resources>views>admin.blade.php

Resources>views>home.blade.php

Resources>view>welcome.blade.php

...

@component('components.who')

@endcomponent

…

**Create views>components>who.blade.php**

@if (Auth::guard('web')->check())

<p class="text-success">

You are Logged In as a <strong>USER</strong>

</p>

@else

<p class="text-danger">

You are Logged Out as a <strong>USER</strong>

</p>

@endif

@if (Auth::guard('admin')->check())

<p class="text-success">

You are Logged In as a <strong>ADMIN</strong>

</p>

@else

<p class="text-danger">

You are Logged Out as a <strong>ADMIN</strong>

</p>

@endif

Building logout functionality for Admin and User Separate

Until now if we logout as user/admin, we automatically logout as admin/user also. Because logout clears out all the sessions.

Default Laravel logout functionality is inside

Illuminate\Foundation\Auth\AuthenticatesUsers;

We inherit all login-logout functionality and all that form here.

We grabbed the showloginForm() and login() function form here and used in our AdminLoginController.php

Coping the logout() from here and use it in LoginController and AdminLoginController0 with some modification

AdminLoignController.php

public function logout()

{

Auth::guard('admin')->logout();

return redirect('/');

}

LoginController.php

\*We are using a different name of logout function so it don’t override the parent class function

public function userLogout()

{

Auth::guard('web')->logout();

return redirect('/');

}

\*flushing the session will log us out of both user and admin

Creating route for our logout functions:

Route::get('/user/logout', 'Auth\LoginController@userLogout')->name('user.logout');

Route::get('/admin/logout', 'Auth\AdminLoginController@logout')->name('admin.logout');

Everyting is using guest middleware in AdminController, so because of this we would not be able to access the logout() function. The logout function is locked into the guest middleware. ie. In order to use logout you have to be the guest(logged out), which is already a logout person.

We will put a little disclaimer here to tell laravel not to apply this middleware on the logout function

AdminLoginController.php

public function \_\_construct(){

$this->middleware('guest:admin', **['except' => ['logout']]**);

}

LoginController.php

public function \_\_construct()

{

$this->middleware('guest')->except(['logout', **'userLogout']**);

}

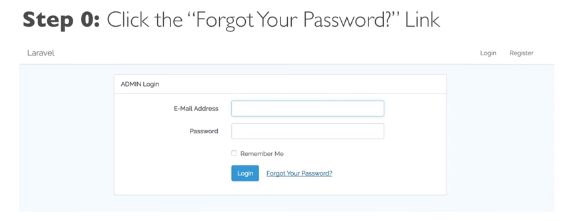
Admin logout link -- localhost:8000/admin/logout

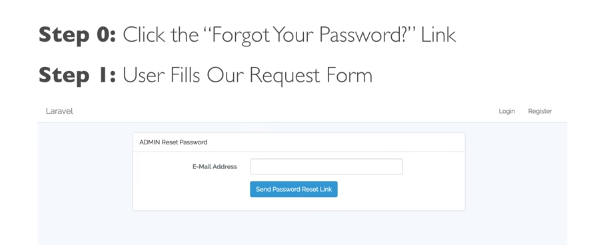
User logout link -- localhost:8000/user/logout

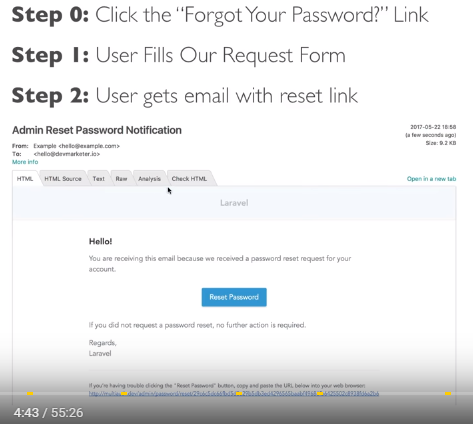
So far in this series we have covered in depth how to set up an authentication system with multiple user types (not to be confused with user roles which are different). In this video of the series we will address questions on how to log users out of the app. First we want to set up some debugging tools so we know if we are logged in as a normal user, as an admin, or both. Once we have that figured out, it will make things a lot easier and now we can set up the logout function. The process is fairly straightforward. We are going to copy the logout() function from the laravel foundation source files. We then can customize it with our guard and finally tie it into a route so it can execute.

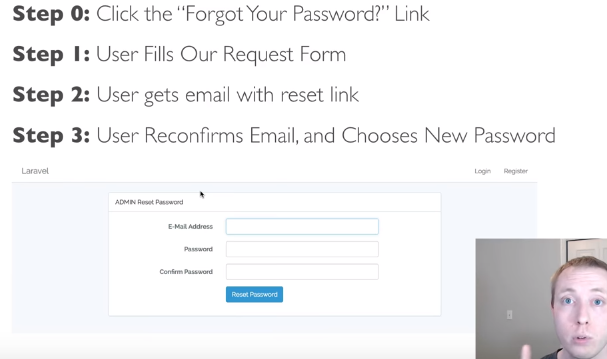
Forgot My Password Functionality across Multiple User Types

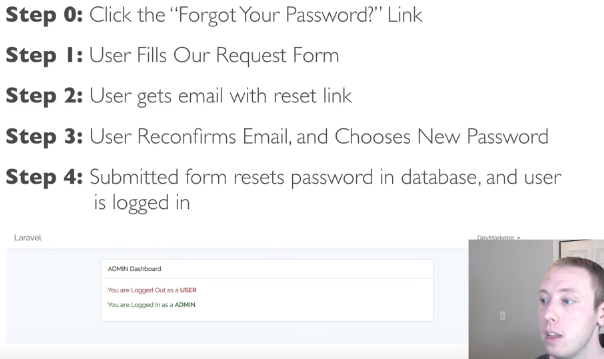
Users perspective for reset my password:



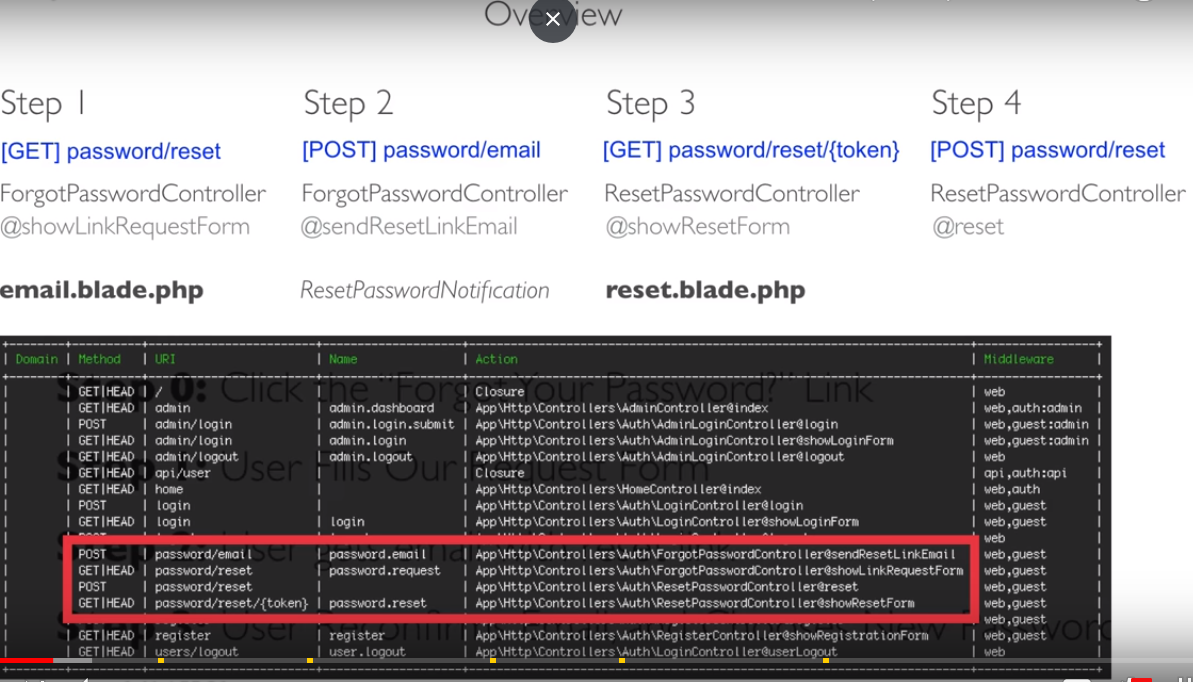








Programming Perspective:



Password reset already work for user, we need to create it for Admin:

Creating Controllers for Admin Password reset:

Duplicating user password reset controllers:-

app>Http>Controllers>Auth>ForgotPasswordController

app>Http>Controllers>Auth>ResetPasswordController

app>Http>Controllers>Auth>AdminForgotPasswordController

app>Http>Controllers>Auth>AdminResetPasswordController

Renaming the classes as AdminForgotPasswordController and AdminResetPasswordController respectively.

>composer dump-autoload

-this refreshes the Laravel cache memory

AdminForgotPasswordController :- uses Password Facade

use Password;

AdminResetPasswordController:- uses Password and Auth Facade and Request

use Illuminate\Http\Request;

use Password;

use Auth;

…

protected $redirectTo = '/admin';

..

Redirecting the user to /admin page after password reset.

…

public function \_\_construct()

{

$this->middleware('guest:admin');

}

..

Using the ‘guest:admin’ i.e guest of admin guard instead of ‘guest’ i.e guest of user guard in both of these Controllers.

If we use ‘guest’ without specifying the guard it will take the guest of default guard i.e- user guard.

**Password Brokers:**

Configuring controllers to work with the admin broker(default they are configured to work with user broker)

AdminForgotPasswordController.php & AdminResetPasswordController.php

protected function broker()

{

return Password::broker('admins');

}

We are using password broker to identify the type of user that this controller has access to.

Config>auth.php

'defaults' => [

'guard' => 'web',

'passwords' => 'users',

],

…

'passwords' => [

'users' => [

'provider' => 'users',

'table' => 'password\_resets',

'expire' => 60,

],

**'admins' => [**

**'provider' => 'admins',**

**'table' => 'password\_resets',**

**'expire' => 15,**

**],**

],

We have two password brokers here ‘users’ and `admins’ here.

AdminResetPasswordController:

We need to define the guard to user when we log the admin in after resetting password:

public function guard()

{

return Auth::guard('admin');

}

Password Reset Routes

Run “php artisna route : list” into terminal take a look of all routes and just copy them over your web.php route file.Admin “Admin” to them so to point Admin Controller.

//Password reset Routes

Route::post('admin/password/email', 'Auth\AdminForgotPasswordController@sendResetLinkEmail')->name('admin.password.email');

Route::get('admin/password/reset', 'Auth\AdminForgotPasswordController@showLinkRequestForm')->name('admin.password.request');

Route::post('admin/password/reset', 'Auth\AdminResetPasswordController@reset')->name('admin.password.update');

Route::get('admin/password/reset/{token}', 'Auth\AdminResetPasswordController@showResetForm')->name('admin.password.reset');

Password Reset Views

Duplicating resources>views>auth>password>email.blade.php and reset.blade.php

email-admin.blade.php

reset-admin.blade.php

Each of these submits to users forgot my password routes, making them to submit to admin forget password routes.

email-admin.blade.php -

<form method="POST" action="{{ route('**admin.password.email'**) }}">

reset-admin.blade.php

<form method="POST" action="{{ route(**'admin.password.update**') }}">

Putting it all together

Step 1:

**Route::get('admin/password/reset', 'Auth\AdminForgotPasswordController@showLinkRequestForm')->name('admin.password.request');**

**Customizing the showLinkRequestForm() so it goes to new view that we created.**

**We gonna copy this code from the source and overide in the AdminForgetPasswordController.**

**Source file -**

**use Illuminate\Foundation\Auth\SendsPasswordResetEmails;**

**AdminForgetPasswordController.php :-**

**..**

public function showLinkRequestForm()

{

return view('auth.passwords.email-admin');

}

…

Step 2:

Route::post('/password/email', 'Auth\AdminForgotPasswordController@sendResetLinkEmail')->name('admin.password.email');

public function sendResetLinkEmail(Request $request)

{

$this->validateEmail($request);

// We will send the password reset link to this user. Once we have attempted

// to send the link, we will examine the response then see the message we

// need to show to the user. Finally, we'll send out a proper response.

$response = $this->broker()->sendResetLink(

$this->credentials($request)

);

return $response == Password::RESET\_LINK\_SENT

? $this->sendResetLinkResponse($request, $response)

: $this->sendResetLinkFailedResponse($request, $response);

}

Going the the source code in **use Illuminate\Foundation\Auth\SendsPasswordResetEmails; we are using the sendResetLinkEmail, this method its going to accept the request form the form we submitted ie the email that is requesting password reset, this method is going to validate this email first , if validation successful the its going to go to the broker that we defined already which broker to use(admin broker).**

**Its going to sendResetLink to which we are passing the email as $request, then it gonna return the response whether it is successful or not.**

**How is the email getting send?**

**sendResetLink is located in Illuminate>Auth>Passwords>PasswordBroker.php**

public function sendResetLink(array $credentials)

{

// First we will check to see if we found a user at the given credentials and

// if we did not we will redirect back to this current URI with a piece of

// "flash" data in the session to indicate to the developers the errors.

$user = $this->getUser($credentials);

if (is\_null($user)) {

return static::INVALID\_USER;

}

// Once we have the reset token, we are ready to send the message out to this

// user with a link to reset their password. We will then redirect back to

// the current URI having nothing set in the session to indicate errors.

$user->sendPasswordResetNotification(

$this->tokens->create($user)

);

return static::RESET\_LINK\_SENT;

}

**sendResetLink is a function of Broker so it is contained inside PasswordBroker.php**

**It is finding the user with that email address and storing it in $user variable(a user model), then it call the method sendPasswordResetNotification() on this $user variable. $user object comes form the user model so it calling this method on user model. In our case the user is Admin so its the Admin model.**

**sendPasswordResetNotification() can be found in Illuminate>Auth>Passwords>canResetPassword.php**

public function sendPasswordResetNotification($token)

{

$this->notify(new ResetPasswordNotification($token));

}

**A token was created and passed in form the PasswordBroker.php function.**

**notify() is the part of Notifiable, this is the official notification that came with Larvel 5.3**

**Now we need to create the notification, thats where we can customize the email and we will customize this funciton to point to our new notification.**

**We need to override this function and create another notification specifically for our admins, we override this method to point to our new notification.**

**Admin model :-**

**…**

**use App\Notifications\AdminResetPasswordNotification;**

**…**

public function sendPasswordResetNotification($token)

{

$this->notify(new AdminResetPasswordNotification($token));

}

**Creating new notificaton AdminResetPasswordNotification**

>php artisan make:notification AdminResetPasswordNotification

Notification created successfully.

**Now we have a new folder inside our app folder Notifications>AdminResetPasswordNotification.php**

**This is actually the email that we are going to send**

**We actually need to prepare if to receive the token, we gonna need the token to prepare the URL for reset password.**

**We are passing the token form the sendPasswordResetNotification() mehtod of our Admin model**

**App>Notificaitons>AdminResetPasswordNotification.php**

…

public $token;

…

\*It needed to be public property to pass it generate the view

public function \_\_construct($token)

{

$this->token = $token;

}

public function toMail($notifiable)

{

return (new MailMessage)

->line('You are receiving this email because we receive a password reset request for your account.')

->action('Reset Password', **route('admin.password.reset', $this->token**))

->line('If you did not request a password request, no further action is required');

}

Step 3:

Route::get('/password/reset/{token}', 'Auth\AdminResetPasswordController@showResetForm')->name('admin.password.reset');

Customizing showResetForm()

This funciton can be found indide.

use Illuminate\Foundation\Auth\ResetsPasswords;

We override it.

public function showResetForm(Request $request, $token = null)

{

return view('**auth.passwords.reset-admin**')->with(

['token' => $token, 'email' => $request->email]

);

}

Step 4:

Route::post('/password/reset', 'Auth\AdminResetPasswordController@reset')->name('admin.password.update');

We don’t have to do anyting actually in this step, just make sure the borker is configured and the guard is configured and the redirect. Step 4 uses these peases of information. Step 4 receives the submission from the view, process it, it saves the updated user information because of the broker and after successfully saving the new password it log us in, it know how to log us in because we define the guard, and it redirect us because of redirectredto after login and we continue on our bussiness.

Change the link of forgot my password in admin-login.blade.php

<a class="btn btn-link" href="{{ route('admin.password.request') }}">

Configure your .env file and test it.

php artisan config:cache

- cache all of your configuration files into a single file using the config:cache Artisan command. This will combine all of the configuration options for your application into a single file which will be loaded quickly by the framework.

It should already work out of the box for the main default user type (as defined in config/auth.php) but we need to do some serious work to get it to do its magic for our other user type. In this video the secondary user type is called "Admins", so that is what you will see here. These are the steps of what we need to do: 1) Duplicate the two controllers that manage our password resets 2) Make sure to match the classnames to the file names for these controllers 3) Set the broker for both of these controllers by creating a protected broker() function 4) Set the middleware for both of these controllers to be the guest middleware FOR OUR GUARD. For example guest:admin. 5) In the ResetPasswordController copy we also need to set the guard in a protected guard() function and also set the redirect url (for the final step of the password reset) by defining the protected $redirectTo property 6) Copy the 4 routes for password resets and make "admin" versions of all of them (we do this by prefixing "admin" to all of the routes) 6) Duplicate the two views in our views/auth/password directory 7) Make sure you update each view's forms to submit to the new admin routes instead of the original routes 8) Back in the ForgotPasswordController we want to override the showLinkRequestForm() method and make sure that it returns the new view we created email-admin.blade.php 9) Now we need to go to our Admin model and override the sendPasswordResetNotification() method and have it use our new notification that we will set up next 10) In the terminal, create a new notification named AdminResetPasswordNotification 11) Open the new notification file and update the text of the email to say what you want. Also create a public property $token and set that property equal to the $token that we pass into it on \_\_construct(). 12) Finally finish the notification by setting the link in the notification equal to our step 3 route and attaching the new token property to the end of the url 13) Now in the ResetPasswordController we need to override showResetForm() method to have it return our view reset-admin.blade.php Yay you are done! Test it out.