Authorization

To not allow one user to edit or delete other users posts at first we need to go to Providers/ServiceAuth.php

We need to make a policy through CLI that is php artisan make:policy BlogPost Policy

write this function

**public function boot()**

**{**

**$this->registerPolicies();**

**Gate::define('update-post',function($user, $post){**

**return $user->id==$post->user\_id;**

**});**

**Gate::define('delete-post',function($user, $post){**

**return $user->id==$post->user\_id;**

**});**

**//**

**}**

**Then go to posts controller**

**In all edit and update**

**$this->authorize('update-post',$post);**

**In delete method**

**$this->authorize('delete-post',$post);**

**This is a short hand helper**

**public function edit($id)**

**{**

**//**

**$post=BlogPost::findorFail($id);**

**// if(Gate::denies('update-post',$post)){**

**// abort(403,"You cant edit this BlogPost");**

**// }**

**$this->authorize('update-post',$post);**

**return view('posts.edit',['post'=>$post]);**

**//This line below is good for a single variable**

**// return view('posts.edit',['post'=>BlogPost::findOrFail($id)]);**

**}**

**Verifying permission of User in php artisan tinker**

**Gate::forUser($user)->denies('update-post',$post);**

**Gate::forUser($user)->allows('update-post',$post);**

**Admin users and overriding permissions**

**At first add admin to Users**

**php artisan make:migration AddAdmintoUsersTable**

**Then go to UserFactory.php**

**Assign 'is\_admin'=>false,**

**then we can assign is\_admin=>true to any specific admin**

**In AuthServiceProvider.php**

**Gate::before(function ($user, $ability) {**

**if ($user->is\_admin && in\_array($ability, ['update-post'])) {**

**return true;**

**}**

**});**

**Admin can authorize and delete any post**

**First we have to create an admin and then**

**Policies Introduction**

**php artisan make:policy BlogPostPolicy --model=BlogPost**

**in BlogpostPolicy.php we have to write**

**return $user->id==$blogPost->id; in update and delete**

**BlogPostPolicy.php**

**in AuthServiceProvider.php**

**public function boot()**

**{**

**$this->registerPolicies();**

**// Gate::define('update-post',function($user, $post){**

**// return $user->id==$post->id;**

**// });**

**// Gate::define('delete-post',function($user, $post){**

**// return $user->id==$post->id;**

**// });**

**//**

**// Gate::define('posts.update','App\Policies\BlogPostPolicy@update');**

**// Gate::define('posts.delete','App\Policies\BlogPostPolicy@delete');**

**//To refactor both codes above we can authorize all functions of BlogPostPolicy.php create,store,update,delete here**

**Gate::resource('posts', 'App\Policies\BlogPostPolicy');**

**//Gate::resource we have defined all as an argument**

**in AuthServiceProvider.php**

**Gate::before(function ($user, $ability) {**

**if ($user->is\_admin && in\_array($ability, ['posts.update','posts.delete'])) {**

**return true;**

**}**

**});**

**}**

**in BlogPostPolicy.php we can simply write**

**public function update(User $user, BlogPost $blogPost)**

**{**

**//**

**// dd('Updating');**

**return $user->id==$blogPost->id;**

**}**

**public function delete(User $user, BlogPost $blogPost)**

**{**

**//**

**// dd('Deleting');**

**return $user->id==$blogPost->id;**

**}**

**Policy or Gate?**

**Gate doesnt specify who should have access to a particular model or resource**

**Instea dof generation posts-update or posts.delete here we can just cretae a Resource controller**

**Now we can just go to $policies in AuthServiceProvider.php**

**'App\BlogPost'=>'App\Policies\BlogPostPolicy'**

**Now we can go to PostsController and simplify it more**

**$this->authorize('delete',$post); instead of**

**$this->authorize('posts.delete',$post);**

**To write it in a much optimal way**

public function destroy(Request $request,$id)

{

//

$post=BlogPost::findorFail($id);

// if(Gate::denies('delete-post',$post)){

// abort(403,"You cant delete this BlogPost");

// }

// $this->authorize('posts.delete',$post);

// $this->authorize('delete',$post);

**$this->authorize($post);-**Most optimal way

//inside create read update every function You can write this

$post->delete();

$request->session()->flash('status','Your post has been deleted');

return redirect()->route('posts.index');

}

**Verifying permissions in bladetemplates**

**To enable a user authenticated to not see**

**index.blade.php**

@can('update',$post);

<p>{{ $post->content }}</p>

<a href="{{ route('posts.edit', [

'post'=>$post->id])}}">Edit</a>

@endcan

delete.blade.php

@can('delete',$post);

<form class="fm-inline" method="POST"action="{{ route('posts.destroy', ['post' => $post->id]) }}">

@csrf

@method('DELETE')

<input class="btn btn-danger" type="submit" value="Delete!" />

</form>

@endcan

**Using middleware to authorize routes**

**We cant use policies for everything**

If we want only the admin to see sepcial messages in a page

**in AuthServiceProvider.php**

Gate::define('home.secret',function($user){

return $user->is\_admin;

});

In contact.blade.php

@can('home.secret')

<p>No details yet</p>

@endcan