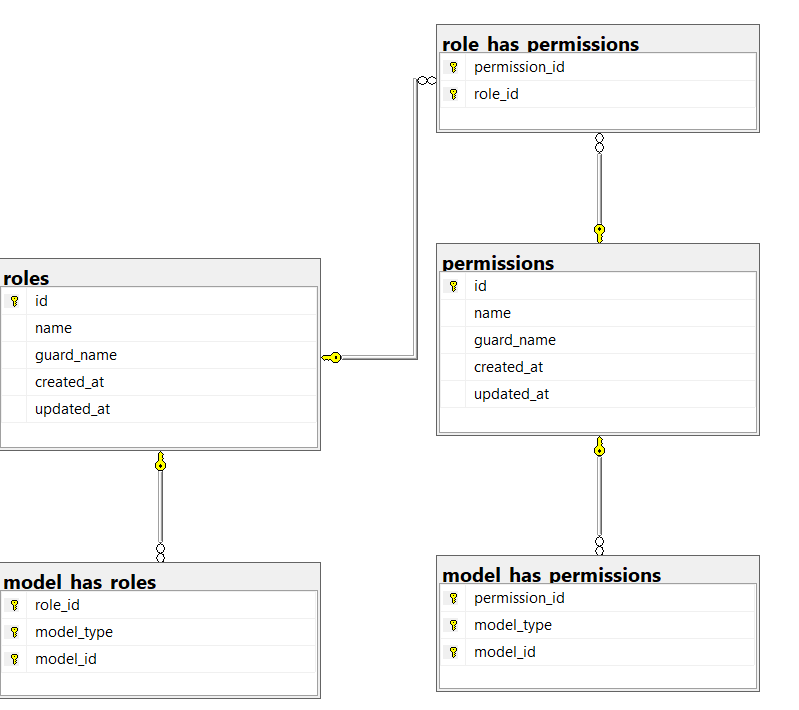
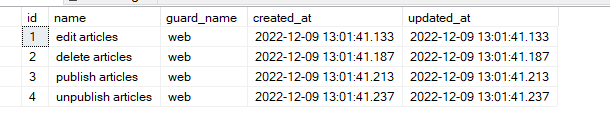
Laravel Spatie – Roles and Permissions

DB Design

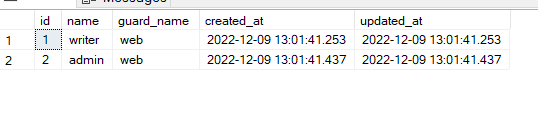


Explanation

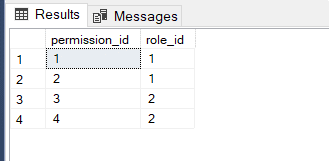
Permissions Table – All the route should be name as a permission and should be in table. For e.g we have create four type of permission as below



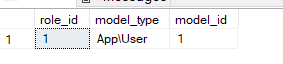
Roles table – All the role can be listed here



role\_has\_permissions – It contains the role and permission mapping. For e.g role id 1 which is writer can (edit / delete) articles Below is the mapping



Now the main mapping is the role id and the user so that we can map any role id to any user schema as below select \* from model\_has\_roles Here model id refers to user\_id of the users table and role id refers to role table. As per below screenshot writer role assigned to user



Users table

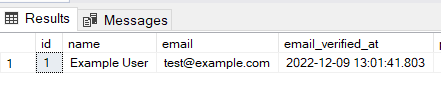


Table model\_has\_permissions not in scope for now

How to setup laravel spatie

Reference Link - <https://www.itsolutionstuff.com/post/laravel-8-user-roles-and-permissions-tutorialexample.html>

Documentation - <https://spatie.be/docs/laravel-permission/v5/introduction>

Giving permission to the user

// Adding permissions to a user

$user->givePermissionTo('edit articles');

// Adding permissions via a role

$user->assignRole('writer');

$role->givePermissionTo('edit articles');

Lets suppose we have user with Name puneet with id 2 and want to create as a Admin

$user = User::where(‘id’,2);

$user->assignRole(‘Admin’);

We need to define middleware for the laravel spatie in kernel.php

'role' => \Spatie\Permission\Middlewares\RoleMiddleware::**class**,

'permission' => \Spatie\Permission\Middlewares\PermissionMiddleware::**class**,

'role\_or\_permission' => \Spatie\Permission\Middlewares\RoleOrPermissionMiddleware::**class**,

Authenticating route

Route::**group**(['middleware' => ['auth']], **function**() {

Route::resource('roles', RoleController::**class**);

Route::resource('users', UserController::**class**);

Route::resource('products', ProductController::**class**);

});

For e.g we have product module and productcontroller we have. We can define the permission mapping to the individual functions as below in the construct method

**function** \_\_construct()

{

$this->middleware('permission:product-list|product-create|product-edit|product-delete', ['only' => ['index','show']]);

$this->middleware('permission:product-create', ['only' => ['create','store']]);

$this->middleware('permission:product-edit', ['only' => ['edit','update']]);

$this->middleware('permission:product-delete', ['only' => ['destroy']]);

}

We can validate user permissions from blade file as well with @can method. Below is the example

@can('role-create')

**<a** **class**="btn btn-success" **href**="{{ route('roles.create') }}"**>** Create New Role**</a>**

@endcan

Another example as below we have the role list view having table with edit / delete button. See below highlighted text. @can(‘role-delete’) which is using to filtering the authorised user can only delete the roles

@can('role-delete')

**<table** **class**="table table-bordered"**>**

**<tr>**

**<th>**No**</th>**

**<th>**Name**</th>**

**<th** **width**="280px"**>**Action**</th>**

**</tr>**

@foreach ($roles as $key => $role)

**<tr>**

**<td>**{{ ++$i }}**</td>**

**<td>**{{ $role->name }}**</td>**

**<td>**

**<a** **class**="btn btn-info" **href**="{{ route('roles.show',$role->id) }}">Show**</a>**

@can('role-edit')

**<a** **class**="btn btn-primary" **href**="{{ route('roles.edit',$role->id) }}">Edit**</a>**

@endcan

@can('role-delete')

{!! Form::open(['method' => 'DELETE','route' => ['roles.destroy', $role->id],'style'=>'display:inline']) !!}

{!! Form::submit('Delete', ['class' => 'btn btn-danger']) !!}

{!! Form::close() !!}

@endcan

**</td>**

**</tr>**

@endforeach

**</table>**