Set Up Authorization Using Policies and Gates for a Multi-Role User System

Objective

Learn how to:

1. **Use Policies** (for model-based authorization).
2. **Use Gates** (for general actions, like admin access).
3. **Manage multi-role permissions** (e.g., `admin`, `editor`, `user`).

---

1. Setup User Roles

First, add a `role` column to users:

```
php artisan make:migration add_role_to_users_table --table=users
```

```
// In the migration file
public function up()
{
    Schema::table('users', function (Blueprint $table) {
        $table->enum('role', ['admin', 'editor', 'user'])-
>default('user');
    });
}
```

Run:

```
php artisan migrate
```

---

2. Option 1: Using Policies (Model-Based Authorization)

Step 1: Generate a Policy

```
php artisan make:policy PostPolicy --model=Post
```

This creates:

```
```

```php
// app/Policies/PostPolicy.php
namespace App\Policies;

use App\Models\User;
use App\Models\Post;

class PostPolicy
{
    public function viewAny(User $user) { return true; }
    public function view(User $user, Post $post) { return true; }
    public function create(User $user) { return $user->role === 'admin'
|| $user->role === 'editor'; }
    public function update(User $user, Post $post) { return $user->role
=== 'admin' || ($user->role === 'editor' && $post->user_id === $user-
>id); }
    public function delete(User $user, Post $post) { return $user->role
=== 'admin'; }
}
```

Step 2: Register the Policy

```php
// app/Providers/AuthServiceProvider.php
protected $policies = [
    Post::class => PostPolicy::class,
];
```

Step 3: Use in Controllers

```php
// app/Http/Controllers/PostController.php
public function edit(Post $post)
{
    $this->authorize('update', $post); // Checks PostPolicy::update()
    return view('posts.edit', compact('post'));
}
```

PROF

Or in **Blade views**:

```blade
@can('update', $post)
    <a href="{{ route('posts.edit', $post) }}">Edit</a>
@endcan
```

---

3. Option 2: Using Gates (General Actions)

Step 1: Define Gates

```
// app/Providers/AuthServiceProvider.php
public function boot()
{
    $this->registerPolicies();

    // Admin gate
    Gate::define('admin', function (User $user) {
        return $user->role === 'admin';
    });

    // Editor gate
    Gate::define('editor', function (User $user) {
        return $user->role === 'editor' || $user->role === 'admin';
    });
}
```

Step 2: Use in Controllers

```
public function dashboard()
{
    if (Gate::allows('admin')) {
        return view('admin.dashboard');
    }
    abort(403);
}
```

Or in **Blade views**:

```
@can('admin')
    <a href="/admin">Admin Panel</a>
@endcan
```

PROF

---

4. Combining Policies + Gates for Multi-Role Systems

Example: Allow admins to edit any post, editors only their own

```
// app/Policies/PostPolicy.php
public function update(User $user, Post $post)
{
    return $user->role === 'admin' ||
```

```
            ($user->role === 'editor' && $post->user_id === $user->id);
}
```

```
// app/Providers/AuthServiceProvider.php
Gate::define('manage-posts', function (User $user) {
    return $user->role === 'admin' || $user->role === 'editor';
});
```

Now, check in **Blade**:

```
@can('manage-posts')
    @can('update', $post)
        <a href="{{ route('posts.edit', $post) }}">Edit</a>
    @endcan
@endcan
```

---

5. Testing Authorization

```
// tests/Feature/PostPolicyTest.php
public function test_admin_can_update_any_post()
{
    $admin = User::factory()->create(['role' => 'admin']);
    $post = Post::factory()->create();

    $this->assertTrue($admin->can('update', $post));
}

public function test_editor_can_update_only_their_posts()
{
    $editor = User::factory()->create(['role' => 'editor']);
    $post = Post::factory()->create(['user_id' => $editor->id]);
    $otherPost = Post::factory()->create();

    $this->assertTrue($editor->can('update', $post));
    $this->assertFalse($editor->can('update', $otherPost));
}
```

PROF

Key Takeaways

✅ **Policies**: Best for **model-specific** permissions (e.g., `Post`, `User`).
✅ **Gates**: Best for **general actions** (e.g., `admin`, `editor` access).
✅ **Multi-Role Systems**: Combine both for granular control.