1v0 Ship v1.0 Faster

Laravel Roles vs Policies: Which One Should You Use?

When building secure applications in **Laravel 12**, you'll quickly run into a question: should I use **roles** or **policies** for access control? Both systems are valid and often used together, but they solve different problems. Understanding their differences helps you pick the right tool for each use case.

In this guide, we'll explore what **roles** are, what **policies** are, show code examples for both, and discuss when to use one over the other. We'll also look at combining them for maximum flexibility, plus how roles can be managed via a UI while policies stay inside code.

1 – What Are Roles?

Roles are labels that group permissions together. For example: `admin`, `editor`, and `user`. Roles are stored in the database and can be assigned dynamically to users, usually using the Spatie Permissions package.

```
// Assign a role
$user->assignRole('admin');

// Check role
if ($user->hasRole('editor')) {
    // allow editing
}Code language: PHP (php)
```

Roles are easy to manage via a UI — admins can assign or revoke them from users without touching code.

2 – What Are Policies?

Policies are classes in Laravel that contain authorization logic for specific models. For example, an `ArticlePolicy` might define whether a user can view, update, or delete a particular article. Unlike roles, policies live in code and can include complex, context-aware logic.

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

use App\Models\User;
use App\Models\Article;

class ArticlePolicy
{
    public function update(User $user, Article $article)
    {
        return $user->id === $article->author_id;
    }

    public function delete(User $user, Article $article)
    {
        return $user->hasRole('admin');
    }
}Code language: PHP (php)
```

Policies are registered in `AuthServiceProvider` and are checked using `$this->authorize()` or `@can` in Blade templates.

```
@can('update', $article)
    <a href="/articles/{{ $article->id }}/edit">Edit</a>
@endcanCode language: HTML, XML (xml)
```

Policies allow fine-grained, per-resource control — something roles alone cannot provide.

3 – Roles vs Policies: The Key Differences

- **Roles** = Who you are. Example: Admin, Editor, User.
- **Policies** = What you can do in context. Example: "Can this user edit *this* article?"
- **Roles** are dynamic and often managed from a UI.
- **Policies** are static and live in your codebase.
- **Roles** work best for global access rules.
- **Policies** work best for model-level or record-level rules.

4 – Combining Roles & Policies

In practice, many apps use **both roles and policies**. For example:

```
// ArticlePolicy
public function delete(User $user, Article $article)
{
    return $user->hasRole('admin') || $user->id ===
$article->author_id;
}Code language: PHP (php)
```

Here, admins can delete any article, while regular users can only delete their own. Roles handle the global rule, while the policy adds the per-resource logic.

5 – UI for Roles, Code for Policies

Roles are perfect for an **admin panel UI**, where non-developers can manage who has access to what features. Policies, on the other hand, are developer territory — they should be written in code because they often require logic that depends on models and business rules.

Example flow:

- Admins log in and assign `roles` to users through a UI (no code changes).
- Policies in code use those roles (and additional checks) to enforce security at the model level.

Wrapping Up

Roles and policies are not competitors — they complement each other. **Use roles** to manage global permissions through a UI, and **use policies** for fine-grained, model-specific rules. Together, they give you a secure and flexible authorization system in Laravel 12.

What’s Next

- Laravel Middleware for Role-Based Route Protection — secure routes with role checks.
- How to Create a Multi-Level Role & Permission System in Laravel — go beyond simple roles.
- Creating a Role-Specific Dashboard in Laravel 12 — customize dashboards for each role.