

[How to Assign Roles to Users Dynamically in Laravel](#)

In modern applications, not every user should have the same level of access. For example, an **admin** should be able to manage users, while an **editor** might only create content. Assigning roles dynamically — either during registration, through admin panels, or via business rules — is essential for a scalable permission system.

In this guide, you'll learn **how to assign roles to users dynamically in Laravel 12** using the **Spatie Permissions package**. We'll explore assigning roles via code, through registration, and by building a user-friendly admin interface. Along the way, we'll cover best practices for keeping your role assignments secure and flexible.

1 - Prerequisites

Before continuing, make sure you've installed and set up [Spatie Laravel-Permission](#). This package provides the `assignRole()` and `syncRoles()` methods we'll use throughout this tutorial.

2 - Assign Roles in Code

The simplest way to assign a role to a user is directly in code. This might be used in a seeder, controller, or event listener.

```
// Assign a role
```

```
$user->assignRole('admin');

// Assign multiple roles
$user->assignRole(['editor','moderator']);

// Replace existing roles
$user->syncRoles('user');
```

`assignRole` adds a role without touching existing roles. `syncRoles` replaces all roles with the new ones. These methods can be used wherever you have access to a `User` instance.

3 - Assign Role on Registration

Often you'll want to assign a default role to every new user, such as `user`. To do this, modify your registration logic.

```
// app/Http/Controllers/Auth/RegisterController.php

use App\Models\User;

protected function create(array $data)
{
    $user = User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => bcrypt($data['password']),
    ]);

    // Assign default role
    $user->assignRole('user');

    return $user;
}
```

}Code language: PHP (php)

This ensures all new users get a base role automatically. Later, admins can upgrade their roles if needed.

4 - Build an Admin UI to Assign Roles

For real applications, you'll want a user interface where admins can assign or change roles dynamically. Let's add a route, controller, and view for this feature.

```
// routes/web.php
use App\Http\Controllers\Admin\UserRoleController;

Route::middleware(['auth', 'role:admin'])->group(function () {
    Route::get('/admin/users/{user}/roles',
        [UserRoleController::class, 'edit'])->name('admin.users.roles.edit');
    Route::put('/admin/users/{user}/roles',
        [UserRoleController::class, 'update'])->name('admin.users.roles.update');
});
```

Code language: PHP (php)

Controller logic:

```
// app/Http/Controllers/Admin/UserRoleController.php
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Spatie\Permission\Models\Role;

class UserRoleController extends Controller
```

```
{
    public function edit(User $user)
    {
        $roles = Role::all();
        return view('admin.users.roles', compact('user','roles'));
    }

    public function update(Request $request, User $user)
    {
        $user->syncRoles($request->roles ?? []);
        return redirect()->back()->with('status','Roles updated
successfully');
    }
}
Code language: PHP (php)
```

Blade view (resources/views/admin/users/roles.blade.php):

```
@extends('layouts.app')

@section('content')
<div class="container">
    <h2>Assign Roles to {{ $user->name }}</h2>

    <form method="POST" action="{{{
route('admin.users.roles.update',$user) }}}">
        @csrf
        @method('PUT')

        @foreach($roles as $role)
            <div class="form-check">
                <input class="form-check-input" type="checkbox" name="roles[]"
value="{{{ $role->name }}}">
                {{{ $user->hasRole($role->name) ? 'checked' : '' }}}>
                <label class="form-check-label">{{{ $role->name }}}</label>
            </div>
        @endforeach

        <button type="submit" class="btn btn-primary mt-3">Save</button>
    </div>
</section>
```

```
</form>
</div>
@endsectionCode language: HTML, XML (xml)
```

This interface allows an admin to dynamically assign or revoke roles from any user by simply checking/unchecking boxes.

5 - Best Practices

- Always validate role assignments to ensure only admins can change them.
- Use `syncRoles()` instead of manually deleting/inserting pivot records — it's safer and cleaner.
- Seed roles and permissions in migrations or seeders for consistent environments.
- Never allow users to assign their own roles during registration — keep role assignment restricted to admin logic or controlled rules.

Wrapping Up

In this guide, we covered how to **assign roles to users dynamically in Laravel 12**. You learned how to assign roles in code, give default roles during registration, and build a UI where admins can update roles with checkboxes. With these tools, your app can grow to support complex role-based access scenarios easily.

What's Next

- [Building a Role-Based Admin Panel in Laravel 12](#) — take your roles further by managing content and dashboards.
- [Creating a User-Friendly Roles & Permissions UI in Laravel](#) — polish the front-end experience.
- [How to Give and Revoke Permissions to Users in Laravel](#) — manage permissions directly from the UI.