I think that's a much better approach, especially in larger projects. But now, we have another problem. Just imagine how many classes we need to create to store a course:

- CreateCourseRequest
- CourseData
- CourseResource
- LessonData
- LessonResource
- A few value objects here and there

And in this example, we have only two models! What if the domain model of this feature is much more complex? You can quickly end up with 10-15 classes to implement the CRUD functionality for courses. It's not the end of the world, but it can be very frustrating. Fortunately, we have an elegant solution. But first, let's summarize what a DTO is:

- It's an object that holds and transfers the data of a model.
- It can be used inside your application between components. Like in the example, when we created a DTO in the controller from a request and passed it to a service.
- But it can also be used outside of your application. So instead of having a request, a resource, and a DTO for the course, why not just have one DTO to rule them all?

Enter the laravel-data package by Spatie. You can use one DTO to act as a:

- Request (with validation rules)
- Resource
- And a simple DTO

Important note: If you want to use DTOs you don't need to go with laravel-data. You can write pure PHP objects, and you'll do just fine. But I find this package so helpful; I cannot imagine a large project without it.

This is what a laravel-data DTO looks like: