a class containing methods used **one time in the entire application**.

- If you don't put every query inside a repository, then why do you even use repositories in the first place? In this case, you'll end up with an inconsistent structure. Some queries are in controllers or models; others live in a repository. In my experience, it's not an optimal solution.
- This is a personal one: I don't feel that `$products->search()` or `$this->products->getById($id)` fits well into Laravel.

However, repositories have some good attributes as well:

- You don't need to have one repository for every model. Let's imagine you have a big enterprise project with 200+ tables. One "module" or feature set is a very basic issue tracker. This feature set requires only six tables and 500 lines of database-related code. You don't have to spread these 500 lines of code in 6 repositories (as you would do with models). You can write only one class called `IssueTrackerRepository`. You cannot do that with models. I think it can be helpful in some situations.

To be honest, that's the only benefit I can think of. If you're already familiar with DDD, you probably heard something like this: *The repository pattern abstracts the data store and enables you to replace your database without changing your business code.* That's true. However, in the last decade, I've faced a lot of strange feature requests, but two of them never came up:

- Nobody ever asked me to change the programming language of an existing project.
- Nobody ever asked me to change the database engine under a running project.

So I'm not authentic to talk about it as an advantage. That being said, I don't think repositories are helpful in most situations. **In fact, Laravel has a better solution: custom query builders.**