Laravel provides you a bunch of handy methods which you can apply on your Eloquent queries to filter down the results.

you run into a situation when you have to reuse some of the conditions more often.

Laravel provides a solution for wrapping your conditions into a readable and reusable statement, called **Scopes**. In this post, I'll show you how to easily integrates scopes into your Laravel models.

Creating Scopes in Laravel

Consider you are building a project management application, and on various places, you have to fetch all completed projects. You can use the following condition to retrieve completed projects.

```
$completedProjects = Project::where('completed', 1)->get();
```

You might need to use the above condition in various places throughout your application. You can use Laravel scopes to DRY up the code. A scope is just a method which you can use in your model to encapsulate the syntax used to execute a query such as above. Scopes are defined by prefixing the name of a method with scope, as below.

```
class Project extends Model
{
    public function scopeCompleted($query)
    {
        return $query->where('completed', 1);
    }

}
```

With the scope defined above, you can execute it like so:

```
$completedProjects = Project::completed()->get();
```

Creating Dynamic Scopes in Laravel

However, if you want to make this scope dynamic and want to get completed on non-completed projects then you can supply an argument, just define an input parameter as you would do on any model method.

```
class Project extends Model {


    public function scopeCompleted($query, $arg)
    {
        return $query->where('completed', $arg);
    }
}
```

With the input parameter defined, you can use the scope like this:

```
// Get completed projects
$completedProjects = Project::completed(1)->get();


// Get incomplete projects
$nonCompletedProjects = Project::completed(0)->get();
```

Using Scopes with Relations

You'll often want to use scopes in conjunction with relations. For instance, you can retrieve a list of projects associated with a user:

```
$user = User::findOrFail(1); // 1 is user id
$completedProjects = $user->projects()->completed(1)->get();
```

Laravel scopes are best to use when you have some repetitive queries and want to reuse the code again and again.