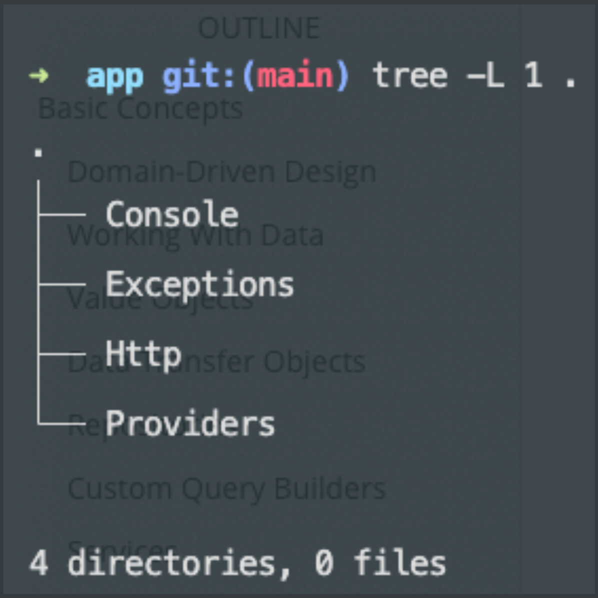
From this array, we can construct the `Database\Factories\Subscriber\SubscriberFactory` class name. Also if you don't like to write `$fillable` arrays you can define `$protected $guarded = [];` in the `BaseModel`.

Applications

As I mentioned in the first section, I will use Inertiajs in the demo application. Please don't panic if you don't know it. The main difference between an Inertiajs and a classic Blade MVC application is one line on the Laravel side. Instead of returning a `view`, you have to return an Inertia specific class from the controller. On the frontend side, it's a Vuejs application, so later in this book, I'll share code snippets from good, old Vuejs components (but, of course, 95.13% of code snippets will show you PHP and Laravel code).

So our main app will be an Inertiajs app, but we also need some APIs (for example, tracking when a user opens an e-mail or clicks on a link). And, of course, we'll have some console commands (like every application has some). Now let's see what the app directory looks like:



```
➜  app git:(main) tree -L 1 .
.
├── Console
├── Exceptions
├── Http
└── Providers

4 directories, 0 files
```

As you can see, it follows the default structure of a new Laravel app. In my opinion, it's important to stay close to the default stack so that Laravel upgrades will be as easy as possible.

If you take a closer look inside the Http directory, you can see there's an Api and a Web folder: