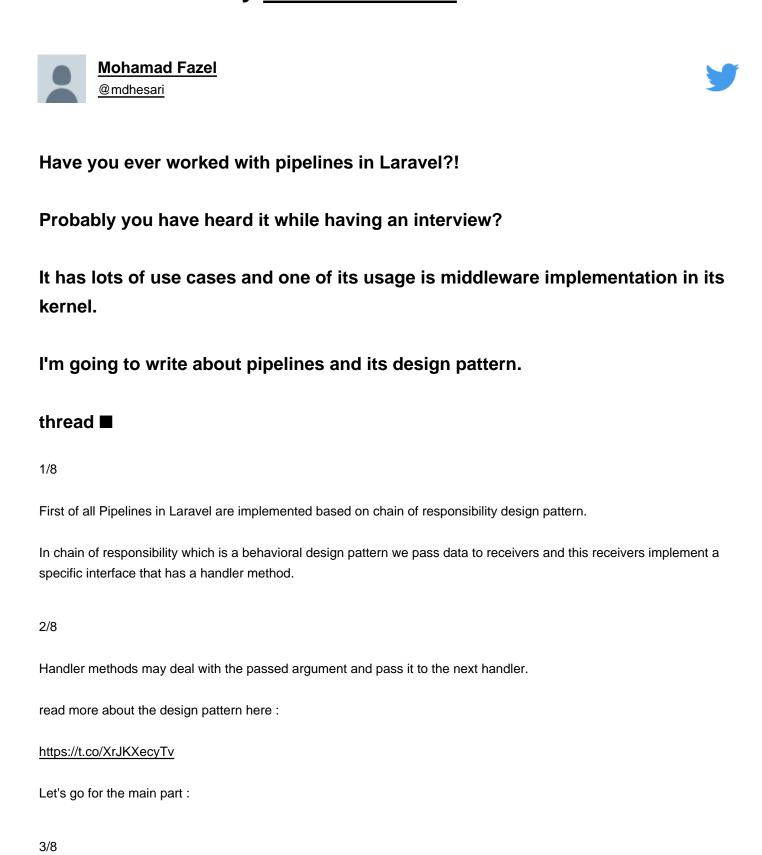
Twitter Thread by Mohamad Fazel



Imagine we want to implement a twitter bot that gets some tweets around the trends and filter the words that we don't wanna show finally send as a new tweet from our bot.

About the filtering part, if it was me, I would implement it with pipelines.

Why?!

4/8

Currently I know exactly what words should be filtered but what about the future?! What if I want to add more filters, I will have to modify the source code and add more switch, if else statements, etc.

I want my code to be robust and maintainable keeping SOLID principles.

5/8

So I'm gonna use pipeline like this (picture):

First I will have my initial filters and also give the ability to myself or other developers to add more filters without even seeing the code.

See the picture and lets go deeper and check the main interface and pipelines ■

6/8

I have defined a contract for all of my filters, every new filter and existing filters must implement this interface and implement the filter method

As you can see in the previous picture we gave our method name "filter" passed to via()

```
namespace App\Filters\Contract;
interface Filterable {
    /**
    * Filter specific text
    *
    * @param string $content
    * @param \Closure $next
    * @return mixed
    */
    public function filter(string $content, \Closure $next);
}
```

7/8

This is one of the filters we have implemented as you can see everything is clear filter method gets an argument called \$content which is the text we wanna filter and when we modify the content and remove bad words it calls the next receiver from pipelines filters we defined

```
<a href="Text-align: left-size: serious seri
```

8/8

That's it all.

This is the pipeline design pattern we have in Laravel.

As you can see there are a lot of advantages using this design pattern having SOLID standards.

But sometimes it's better to keep things simple when our program is not that big and you are alone...