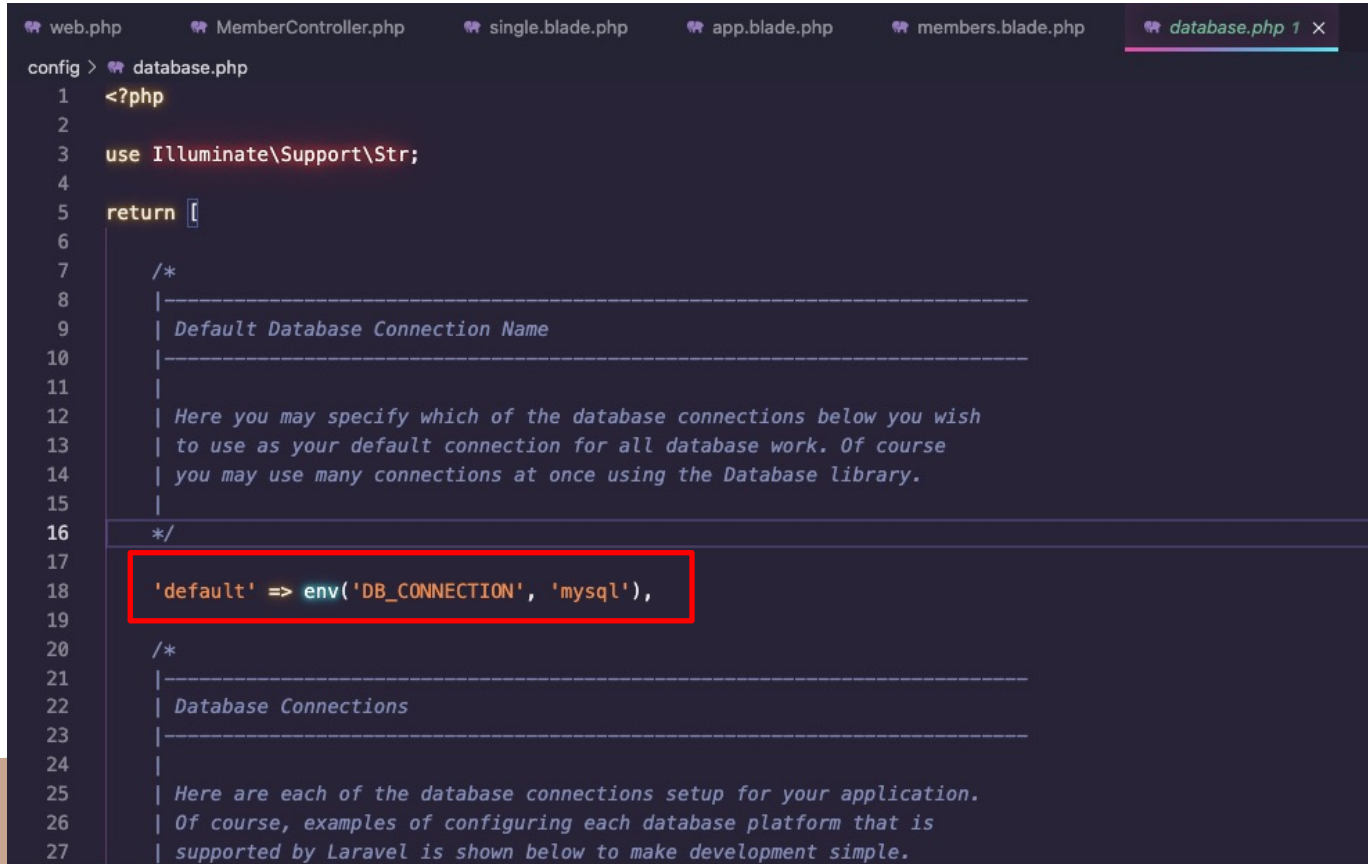




# Database

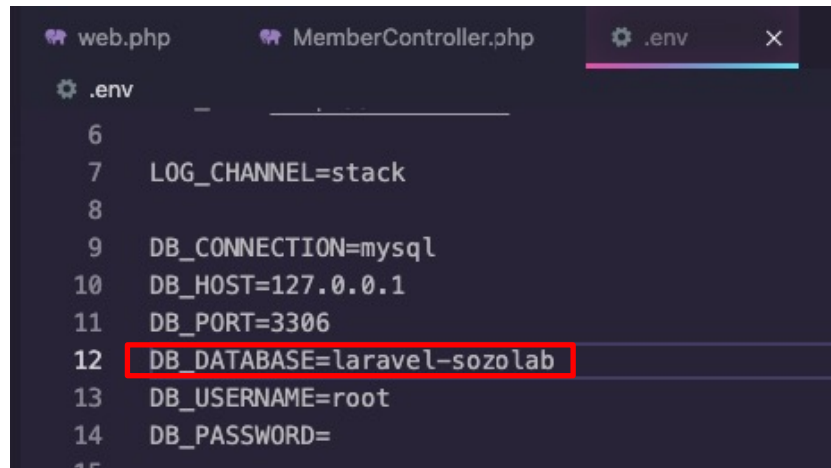
**In a dynamic website, of course, we will not use array data in the script controller, but a database.**

To see the database settings we can look at config/database.php



```
web.php MemberController.php single.blade.php app.blade.php members.blade.php database.php 1 X
config > database.php
1  <?php
2
3  use Illuminate\Support\Str;
4
5  return [
6
7      /*
8       |-----
9       | Default Database Connection Name
10      |-----
11      |
12      | Here you may specify which of the database connections below you wish
13      | to use as your default connection for all database work. Of course
14      | you may use many connections at once using the Database library.
15      |
16      */
17      'default' => env('DB_CONNECTION', 'mysql'),
18
19      /*
20      |-----
21      | Database Connections
22      |-----
23      |
24      | Here are each of the database connections setup for your application.
25      | Of course, examples of configuring each database platform that is
26      | supported by Laravel is shown below to make development simple.
27      |
```

It turns out that laravel uses a variable named .env to set the system environment.

A screenshot of a code editor with a dark theme. The top of the editor shows three tabs: 'web.php', 'MemberController.php', and '.env'. The '.env' tab is active and highlighted with a pink and blue border. Below the tabs, the content of the '.env' file is displayed. It contains several lines of configuration: 'LOG\_CHANNEL=stack', 'DB\_CONNECTION=mysql', 'DB\_HOST=127.0.0.1', 'DB\_PORT=3306', 'DB\_DATABASE=laravel-sozolab', 'DB\_USERNAME=root', and 'DB\_PASSWORD='. The line 'DB\_DATABASE=laravel-sozolab' is highlighted with a red rectangular box. Line numbers 6 through 15 are visible on the left side of the code block.

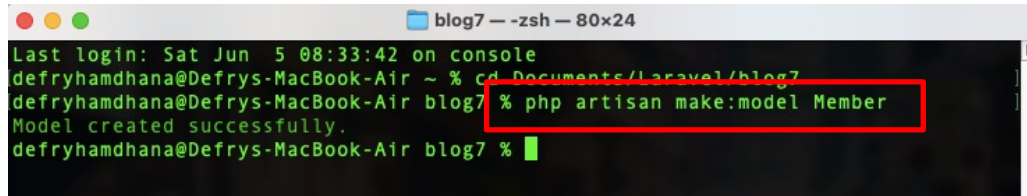
```
.env
6
7 LOG_CHANNEL=stack
8
9 DB_CONNECTION=mysql
10 DB_HOST=127.0.0.1
11 DB_PORT=3306
12 DB_DATABASE=laravel-sozolab
13 DB_USERNAME=root
14 DB_PASSWORD=
15
```

For now we just need to focus on the name of the database we want to sync.

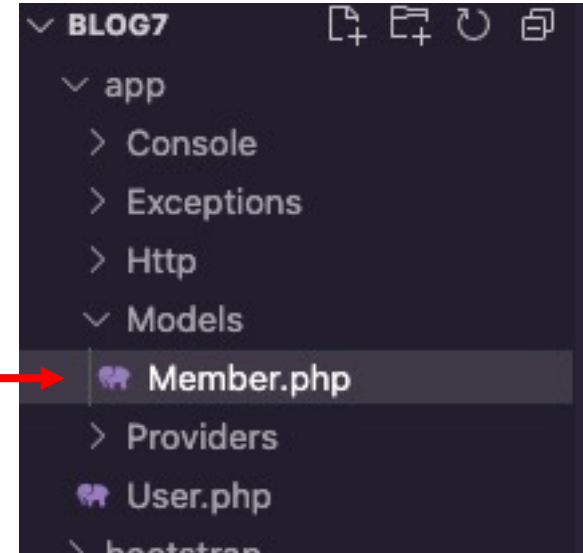
I have created the same data as the previous example in MySQL. And now we will create a Model to connect the database that we have created with our laravel blog.

To create a model, we can use the help of composer with the command: `php artisan make:model NameModel`

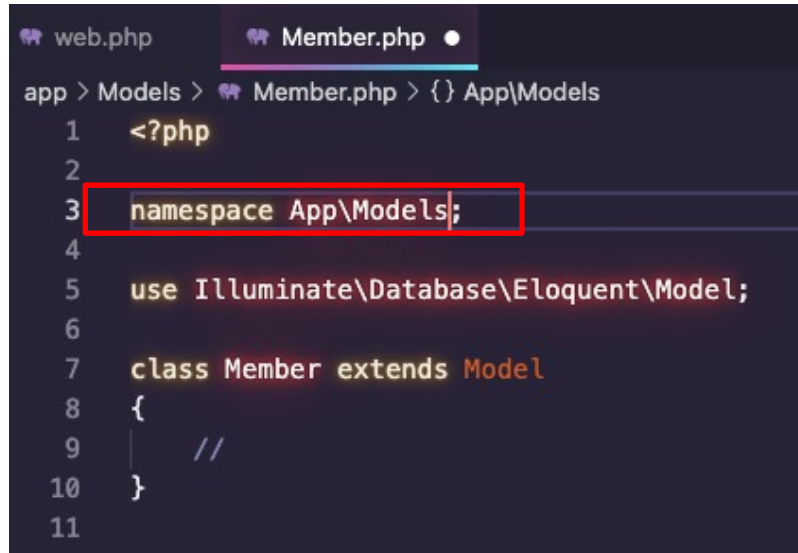
NB: The model will be created in the app folder, to make it more easier I created a special folder with the name Models.



```
blog7 - zsh - 80x24
Last login: Sat Jun  5 08:33:42 on console
defryhamdhana@Defrys-MacBook-Air ~ % cd Documents/Laravel/blog7
defryhamdhana@Defrys-MacBook-Air blog7 % php artisan make:model Member
Model created successfully.
defryhamdhana@Defrys-MacBook-Air blog7 %
```



Don't forget, because the Member.php (model) that we just created we put in the Models folder, then there is a little extra in the path namespace.



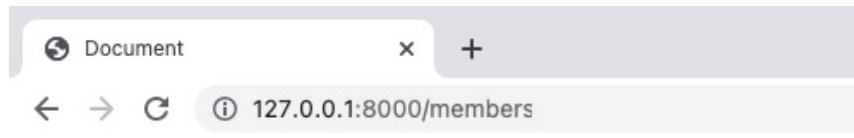
The screenshot shows a code editor with two tabs: 'web.php' and 'Member.php'. The 'Member.php' tab is active and highlighted with a blue bar. Below the tabs, the breadcrumb path 'app > Models > Member.php > {} App\Models' is displayed. The code content is as follows:

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Member extends Model
8  {
9      //
10 }
11
```

The line 'namespace App\Models;' on line 3 is highlighted with a red rectangular box.

Now we can go to MemberController to replace the array data that we have created in mySQL database.

```
web.php Member.php MemberController.php x
app > Http > Controllers > MemberController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Member;
6  use Illuminate\Http\Request;
7
8  class MemberController extends Controller
9  {
10     public function index()
11     {
12         $informations = Member::all();
13         return view('members.members', compact('informations'));
14     }
15
16     public function show($slug)
17     {
18         return view('members.single', ['title' => $slug]);
19     }
20 }
21
```



## **This is a sozolab members page**

**Name: Christina**

Hobby: gardening

**Name: Defry**

Hobby: badminton

**Name: Fikry**

Hobby: football

**Name: Nazmun**

Hobby: reading

**Name: Tan**

Hobby: swimming



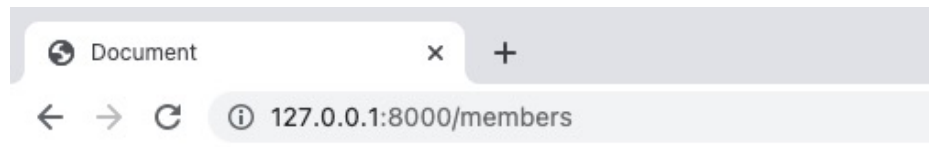
Then what if the data we have in the database is large enough? We will divide the displayed data with pagination.

Let's back to MemberController.php

```
public function index()
{
    $informations = Member::paginate(3);
    return view('members.members', compact('informations'));
}
```

After that, we also add the pagination link in the members view.

```
11 <div>
12     {{ $informations -> links() }}
13 </div>
```



# This is a sozolab members page

**Name: Christina**

Hobby: gardening

**Name: Defry**

Hobby: badminton

**Name: Fikry**

Hobby: football

- <
- 1
- [2](#)
- >

Thank you 😊

