

Lumen, controllers, databaser,
modeller, m.m.

Dagens agenda

- Vi kommer att blanda teori med praktik
 - Så var frågvisa!
- Repetition
- Del 1: Översikt
- Del 2: Controllers
- Del 3: Databaser
 - A) Designa en databas
 - B) Använda migrations för att skapa databasen
 - C) Använda seeds för att fylla databasen
 - D) Läs / Skriv till vår databas
- Del 4: Modeller & ORM

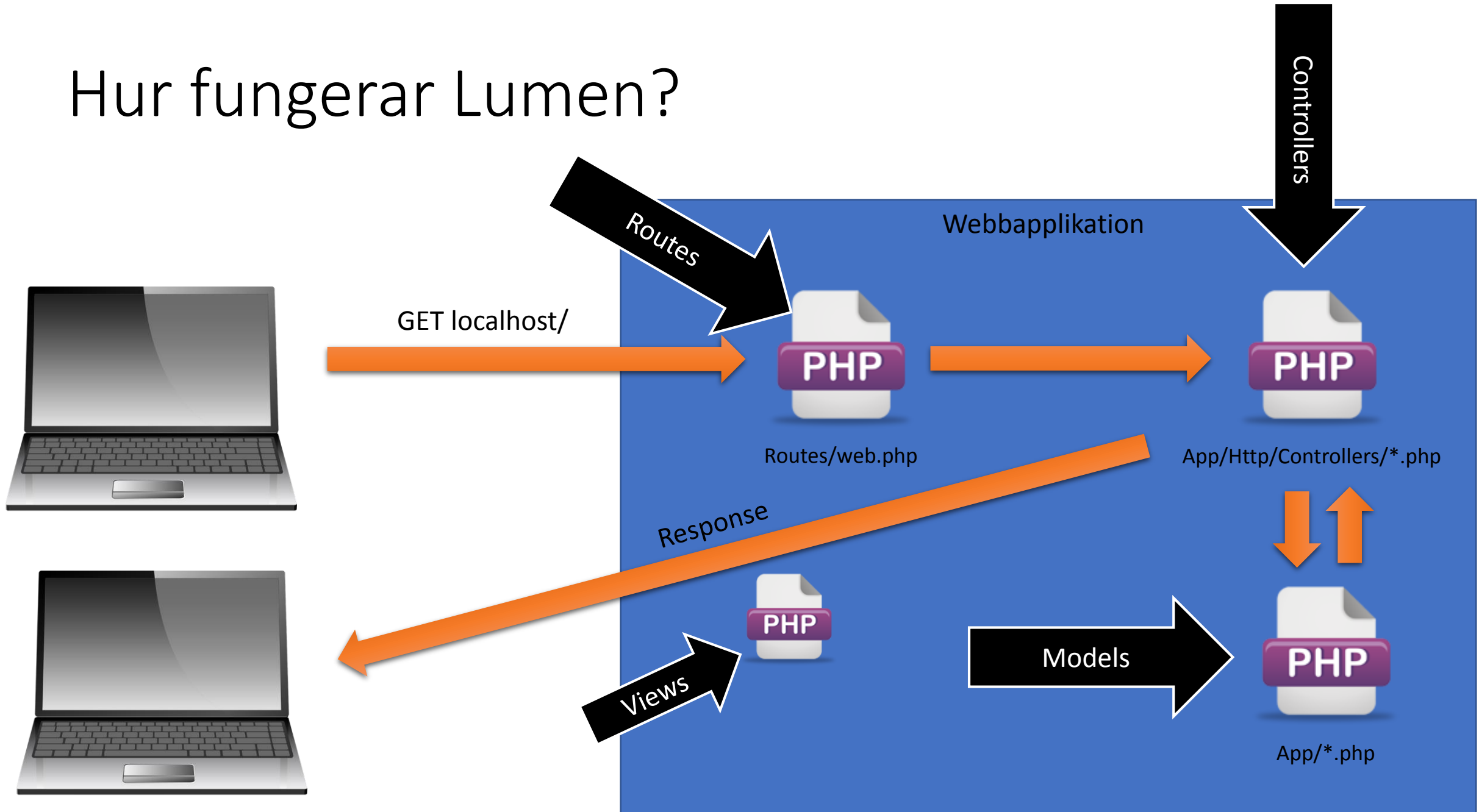


Lumen.

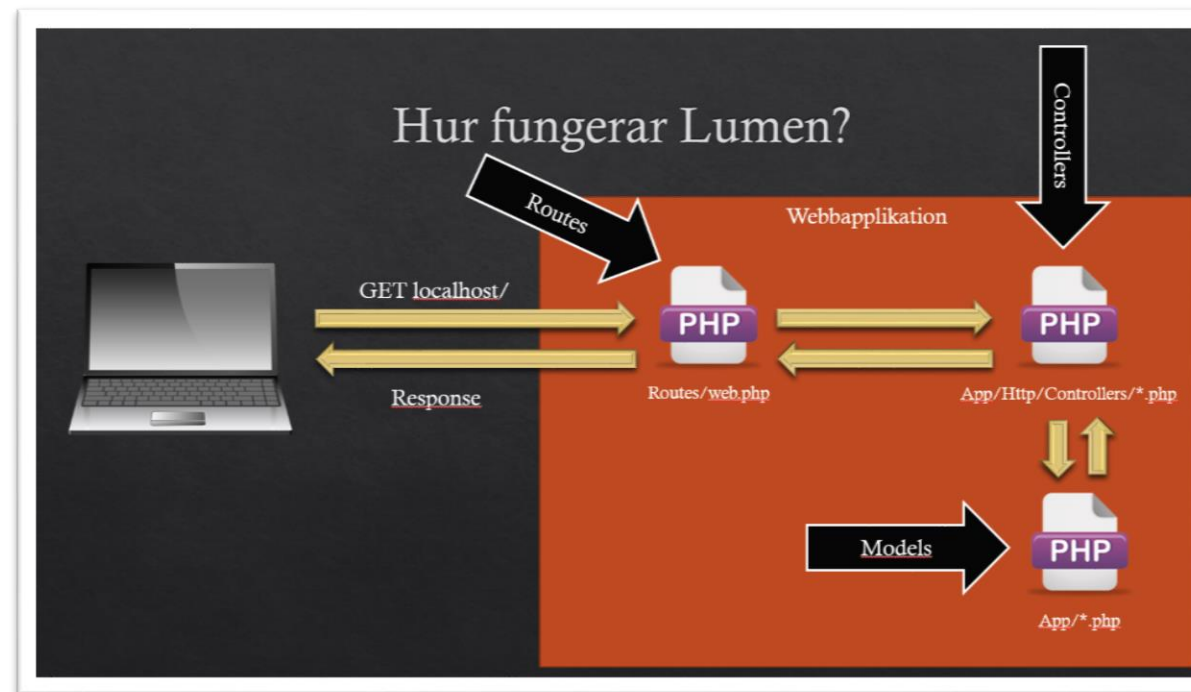
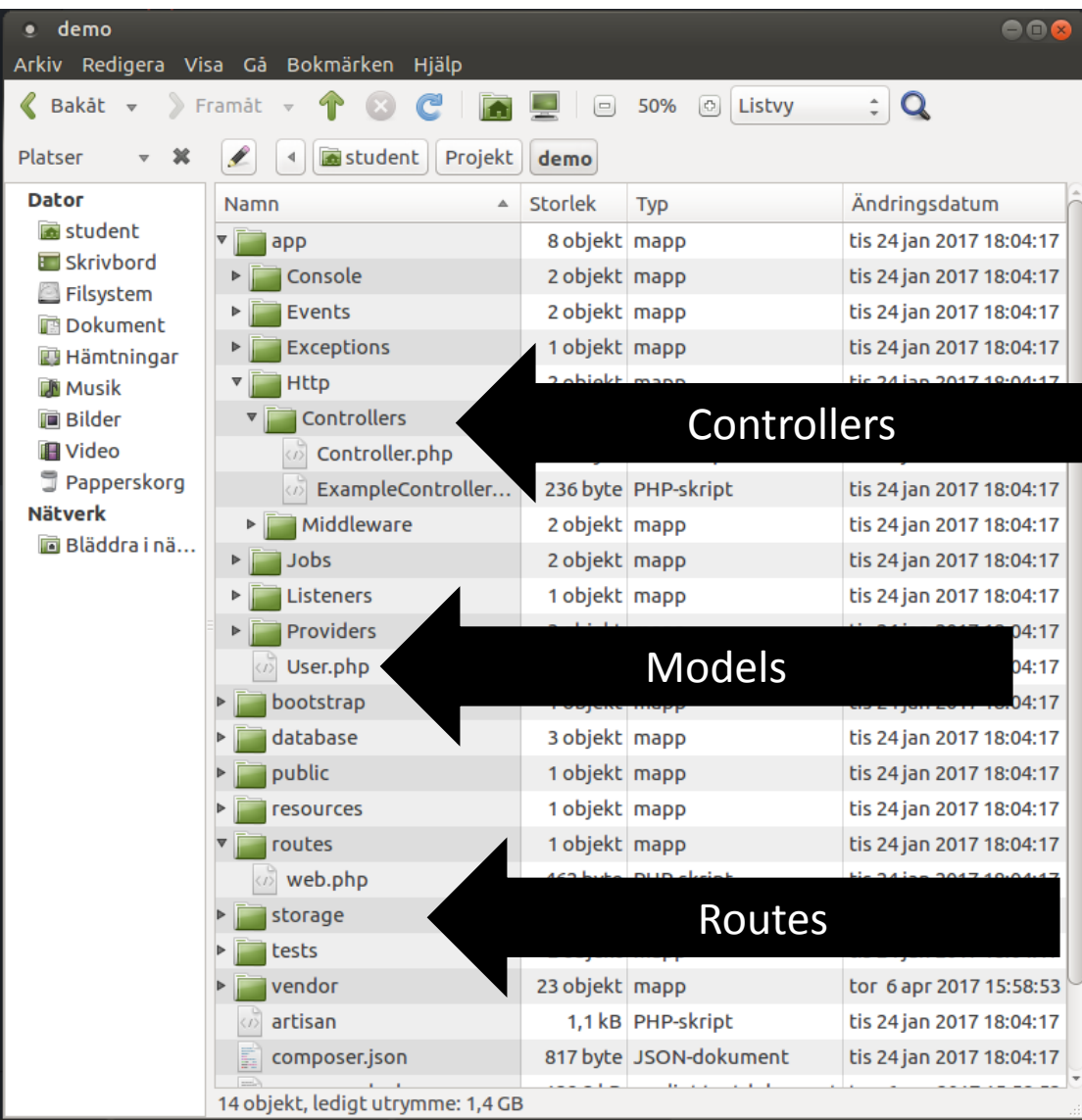
The stunningly fast micro-framework by Laravel.

```
1 <?php
2
3 /**
4  * Reimagine what you expect...
5  */
6 $app->get('/', function() {
7     return ['version' => '5.3']
8 });
9
10 /**
11  * From your micro-framework...
12  */
13 $app->post('framework/{id}', function($framework) {
14
```

Hur fungerar Lumen?



Hur fungerar Lumen? (2)



Routes (Routes/web.php)

Basic Routing

You will define all of the routes for your application in the `routes/web.php` file. The most basic Lumen routes simply accept a URI and a `Closure` :

```
$app->get('foo', function () {  
    return 'Hello World';  
});  
  
$app->post('foo', function () {  
    //  
});
```

Routes (Routes/web.php)

Available Router Methods

The router allows you to register routes that respond to any HTTP verb:

```
$app->get($uri, $callback);  
$app->post($uri, $callback);  
$app->put($uri, $callback);  
$app->patch($uri, $callback);  
$app->delete($uri, $callback);  
$app->options($uri, $callback);
```

Route Parameters

Required Parameters

Of course, sometimes you will need to capture segments of the URI within your route. For example, you may need to capture a user's ID from the URL. You may do so by defining route parameters:

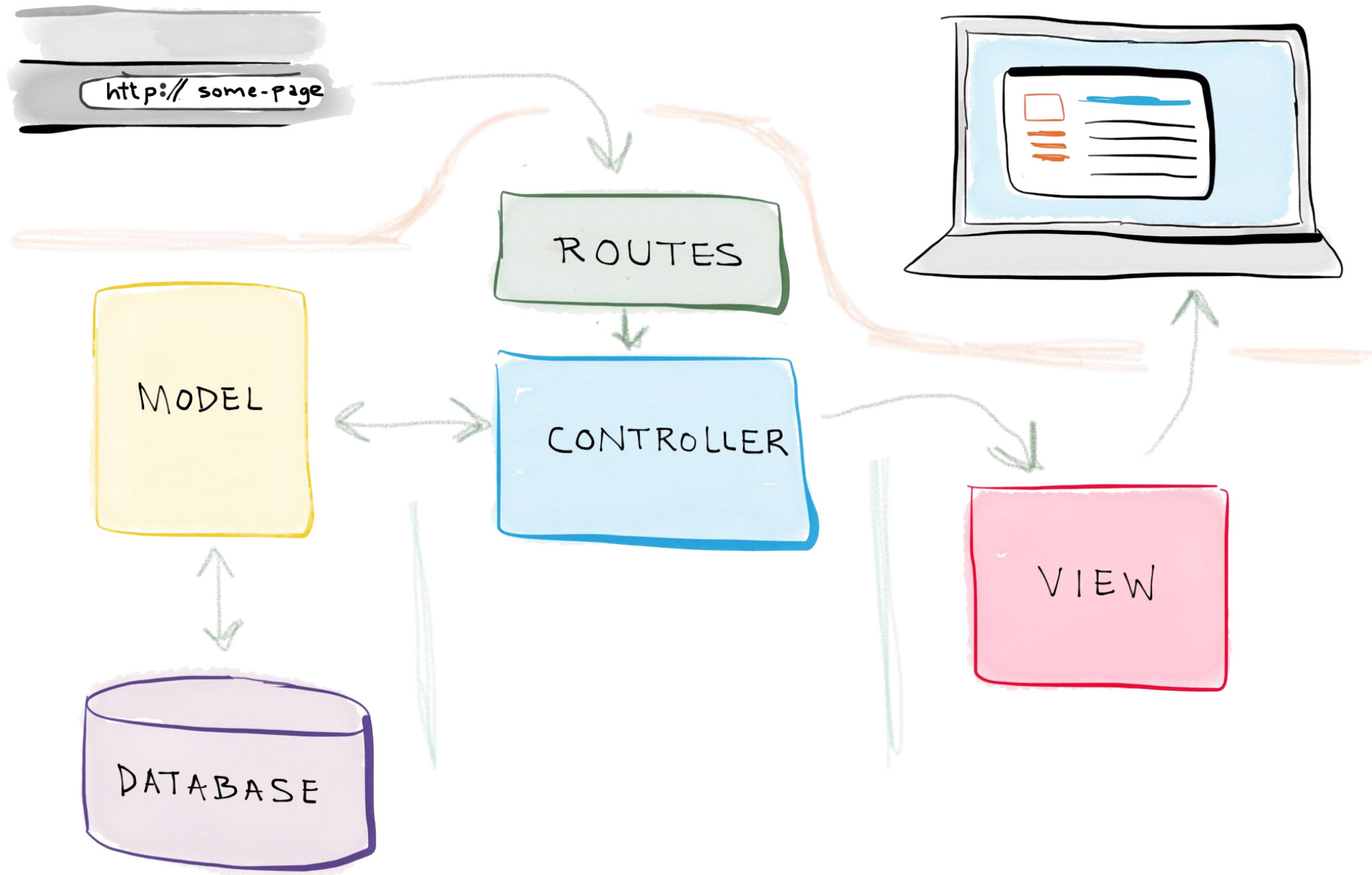
```
$app->get('user/{id}', function ($id) {  
    return 'User ' . $id;  
});
```

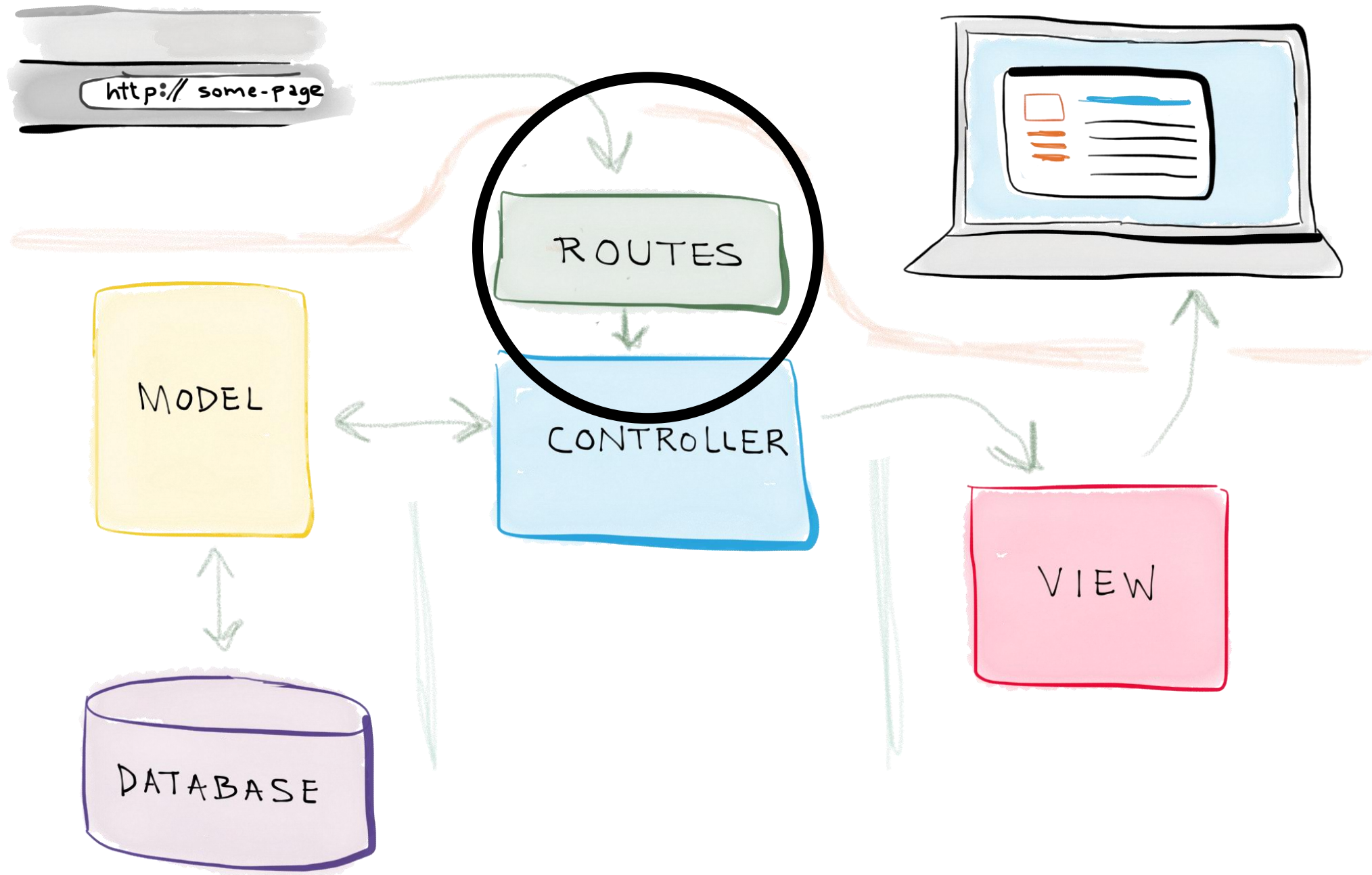
You may define as many route parameters as required by your route:

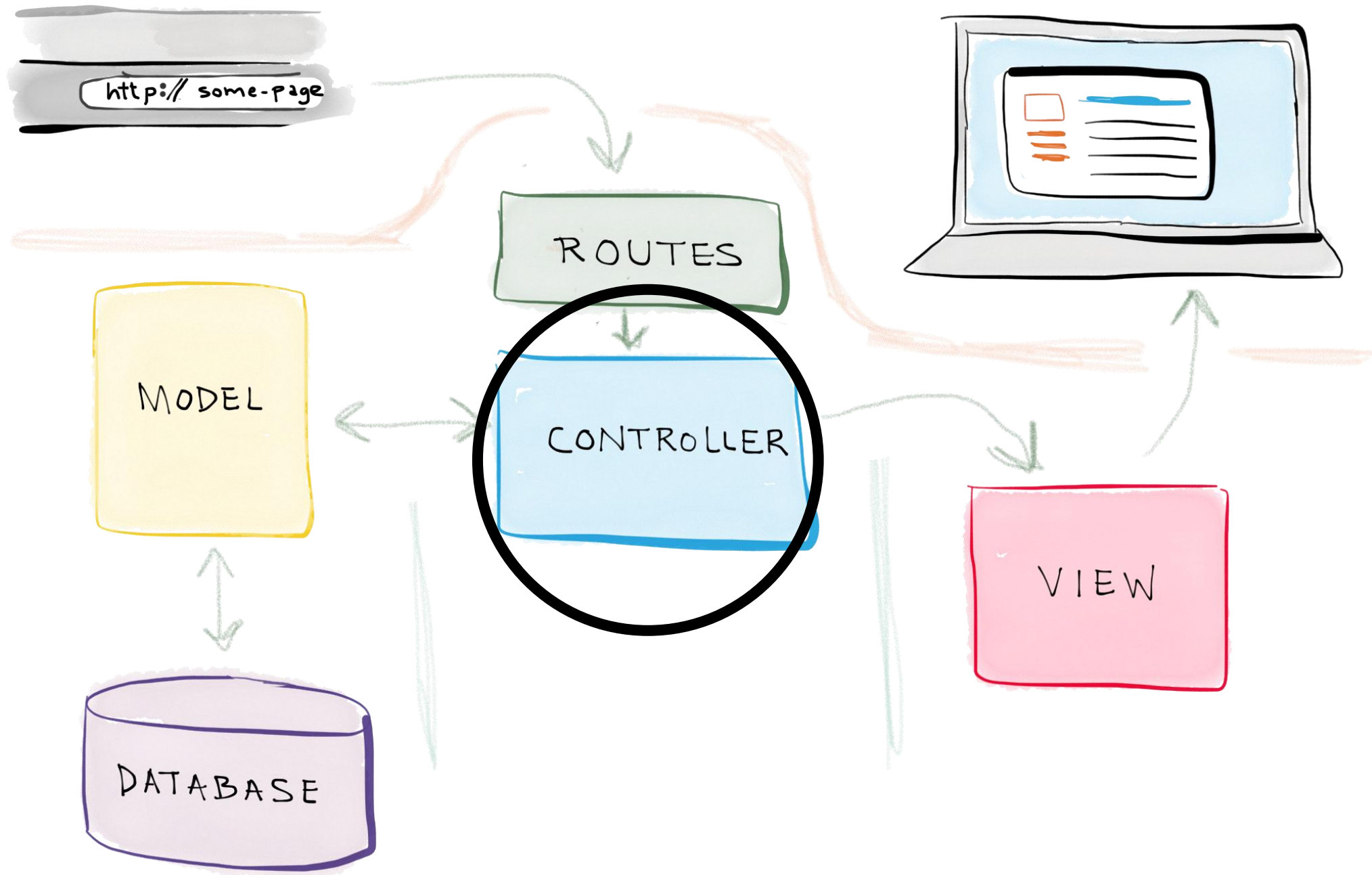
```
$app->get('posts/{post}/comments/{comment}', function ($postId, $commentId) {  
    //  
});
```

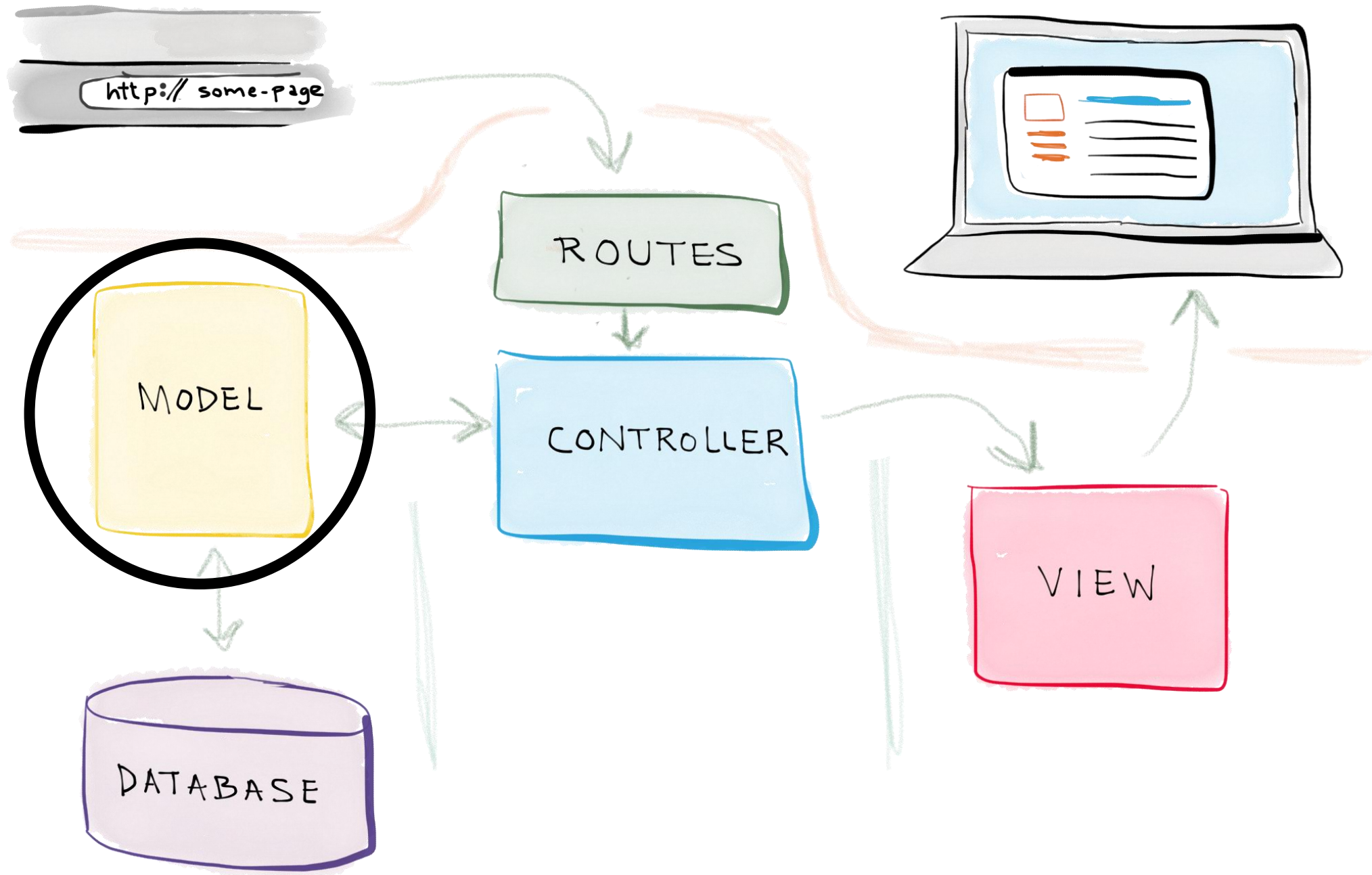
Route parameters are always encased within "curly" braces. The parameters will be passed into your route's `Closure` when the route is executed.

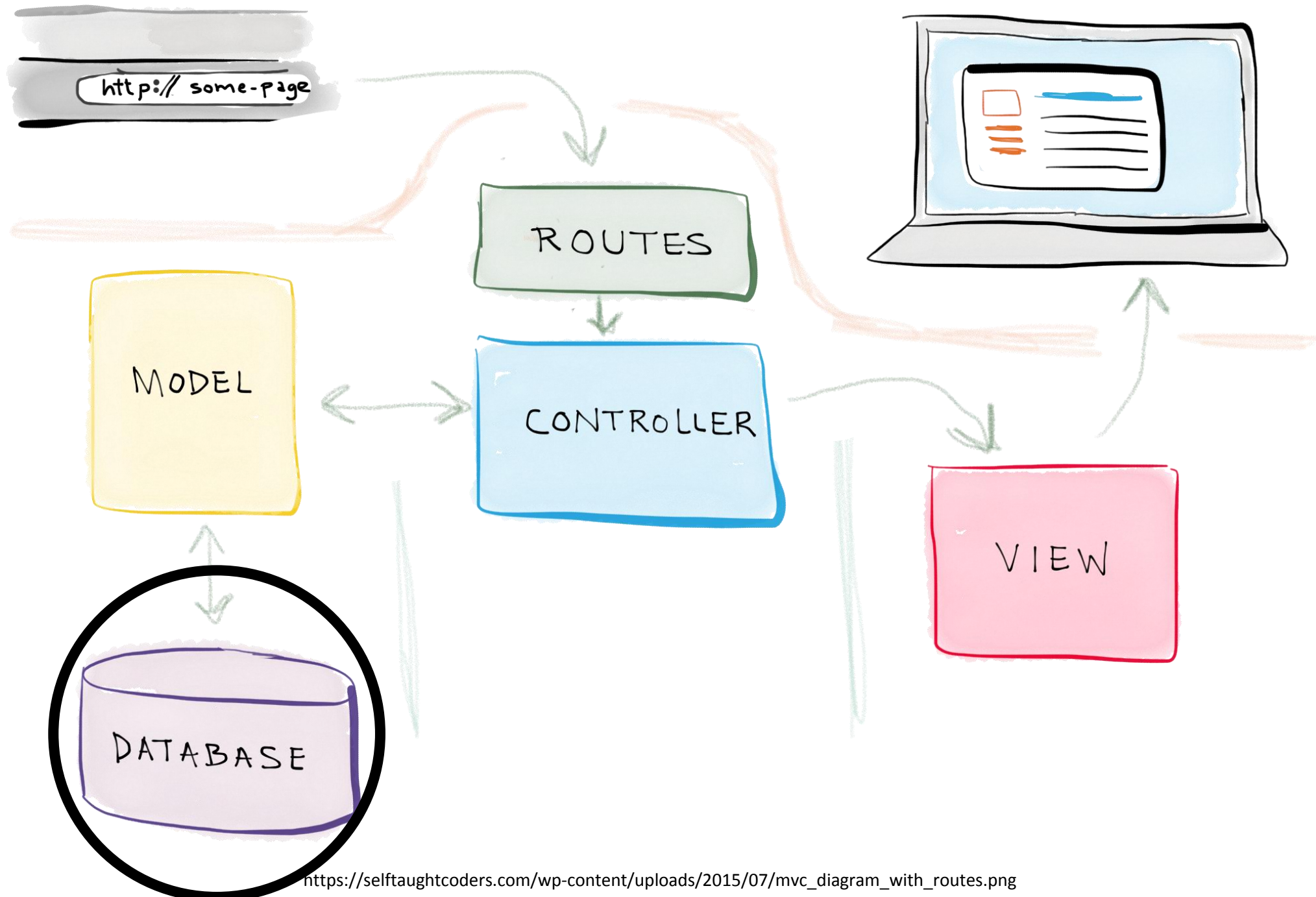
Note: Route parameters cannot contain the `-` character. Use an underscore (`_`) instead.

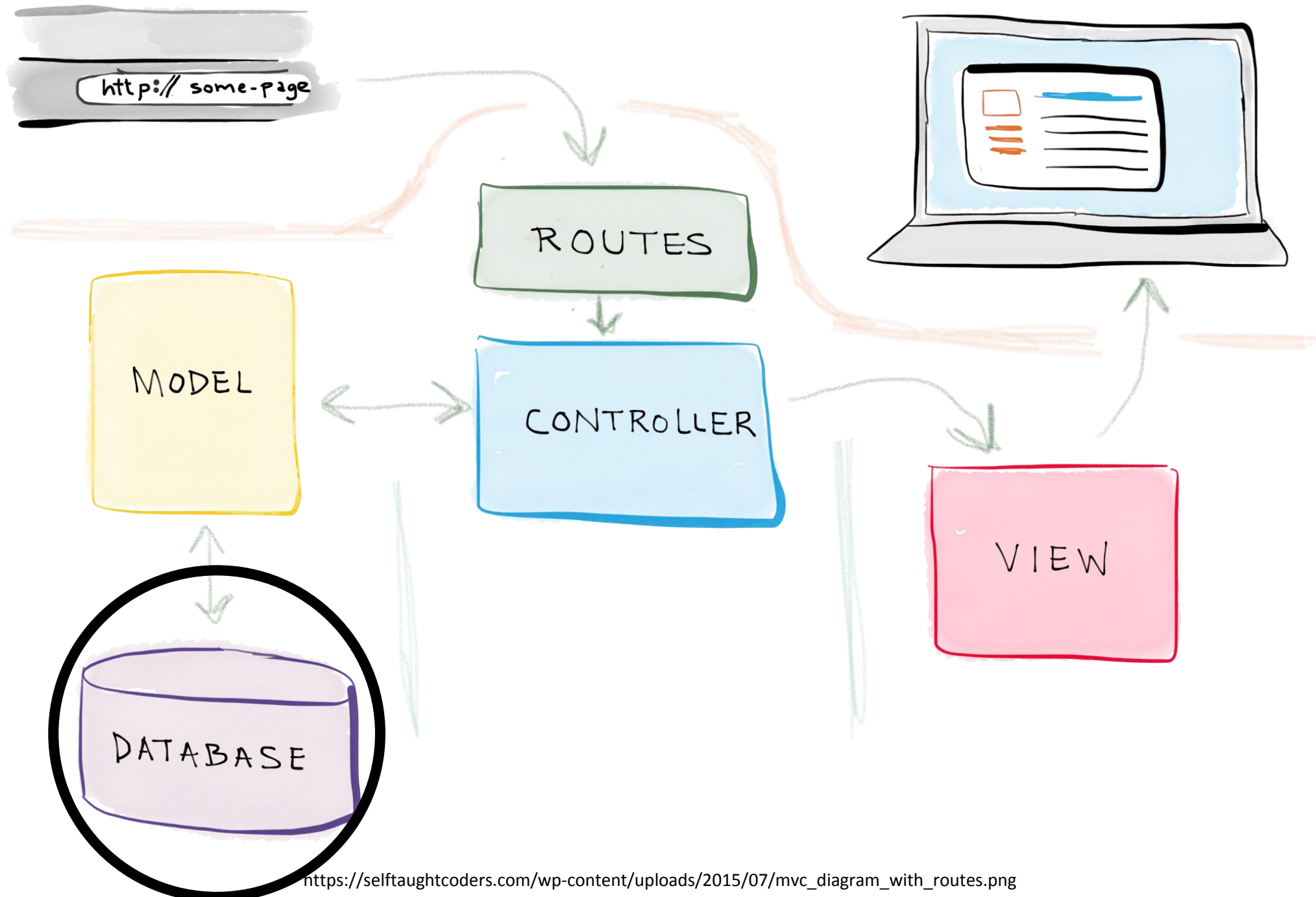


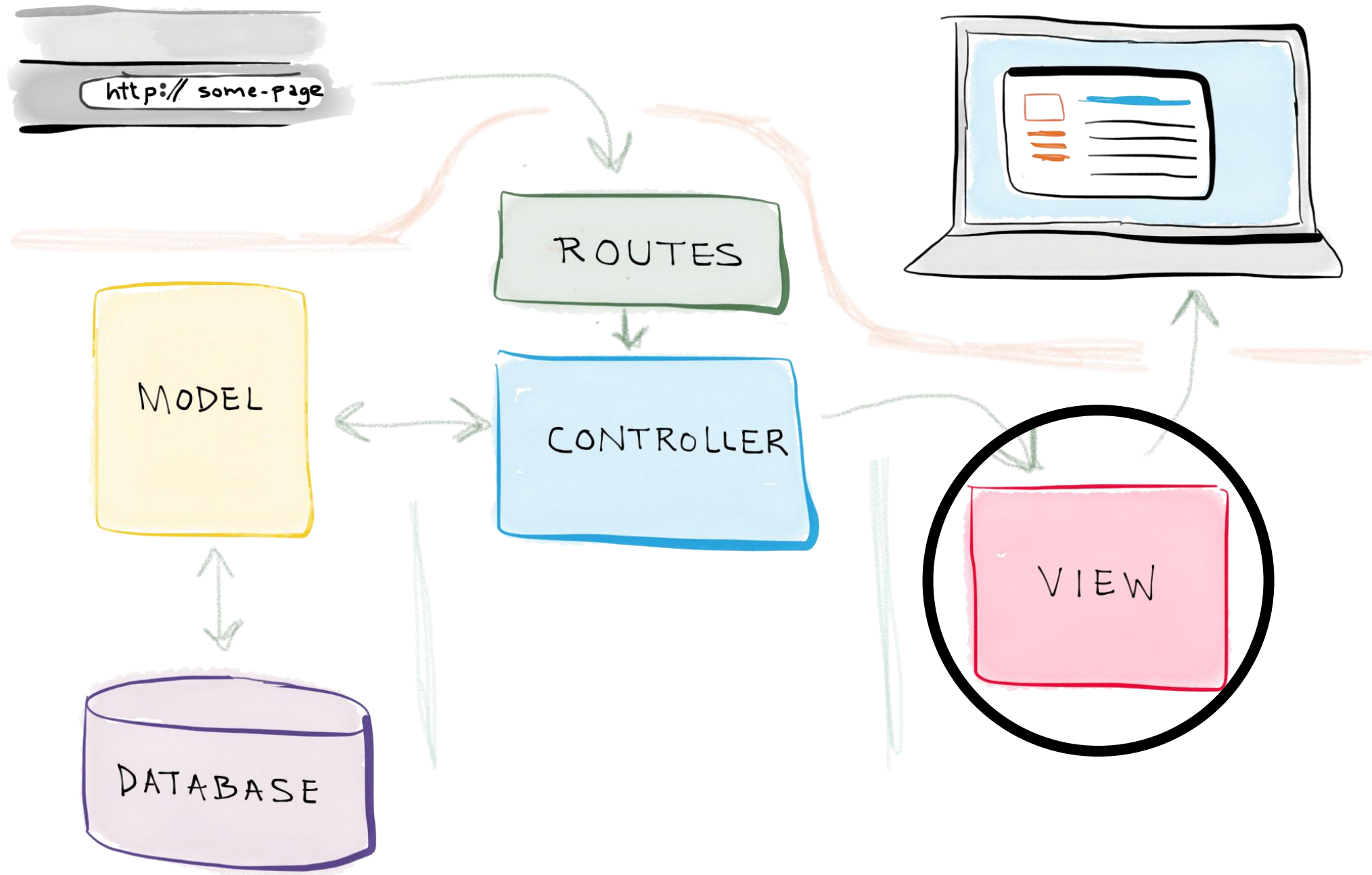












demo

Arkiv Redigera Visa Gå Bokmärken Hjälp

Bakåt Framåt 50% Listvy

Platser student Projekt demo

Dator

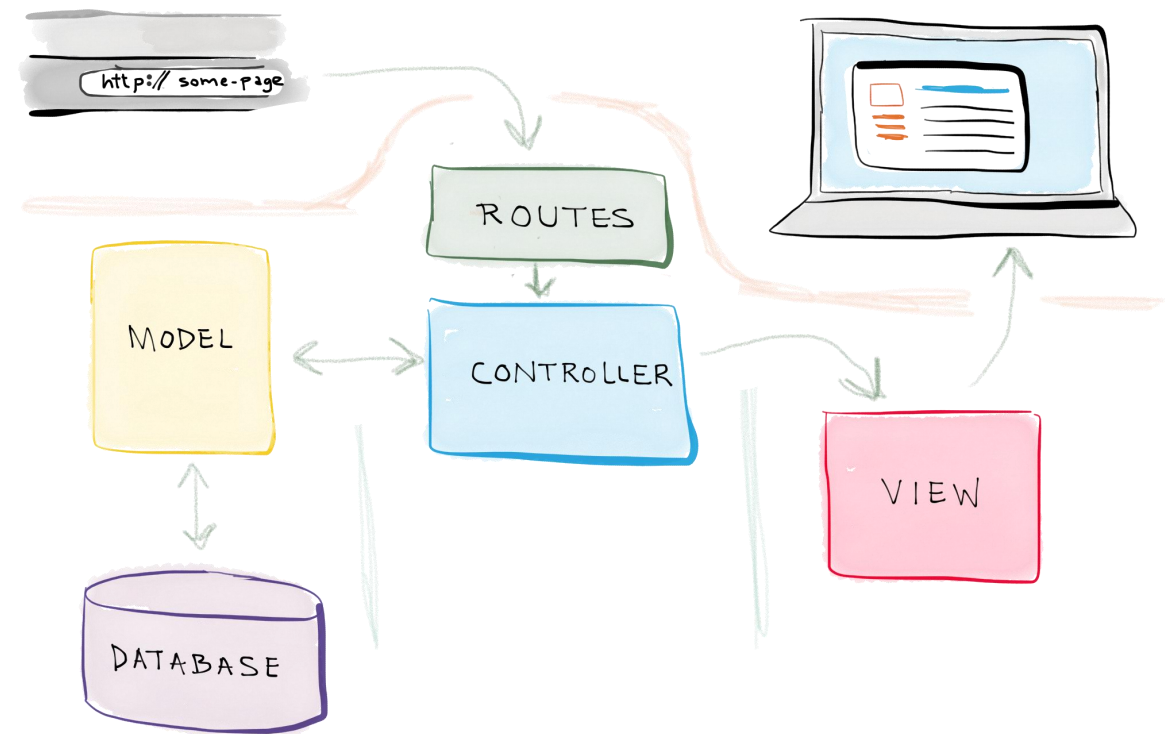
- student
- Skrivbord
- Filsystem
- Dokument
- Hämtningar
- Musik
- Bilder
- Video
- Papperskorg

Nätverk

- Bläddra i nä...

Namn	Storlek	Typ	Ändringsdatum
app	8 objekt	mapp	tis 24 jan 2017 18:04:17
Console	2 objekt	mapp	tis 24 jan 2017 18:04:17
Events	2 objekt	mapp	tis 24 jan 2017 18:04:17
Exceptions	1 objekt	mapp	tis 24 jan 2017 18:04:17
Http	2 objekt	mapp	tis 24 jan 2017 18:04:17
Controllers			
Controller.php			
ExampleController.php	236 byte	PHP-skript	tis 24 jan 2017 18:04:17
Middleware	2 objekt	mapp	tis 24 jan 2017 18:04:17
Jobs	2 objekt	mapp	tis 24 jan 2017 18:04:17
Listeners	1 objekt	mapp	tis 24 jan 2017 18:04:17
Providers	2 objekt	mapp	tis 24 jan 2017 18:04:17
User.php			
bootstrap			
database	3 objekt	mapp	tis 24 jan 2017 18:04:17
public	1 objekt	mapp	tis 24 jan 2017 18:04:17
resources	1 objekt	mapp	tis 24 jan 2017 18:04:17
routes	1 objekt	mapp	tis 24 jan 2017 18:04:17
web.php			
storage			
tests	2 objekt	mapp	tis 24 jan 2017 18:04:17
vendor	23 objekt	mapp	tor 6 apr 2017 15:58:53
artisan	1,1 kB	PHP-skript	tis 24 jan 2017 18:04:17
composer.json	817 byte	JSON-dokument	tis 24 jan 2017 18:04:17

14 objekt, ledigt utrymme: 1,4 GB



Routes

=> Controller => View

```
1 <?php
2
3 use Illuminate\Http\Request;
4
5 $app->get('/', function () use ($app) {
6     return "Hello World!";
7 });
8
9 $app->get('/movies', 'MoviesController@index');
10 $app->get('/movies/{id}', 'MoviesController@show');
11 $app->post('/movies', 'MoviesController@create');
12
13 $app->put('/movies/{id}', function($id){
14     return "Will update the movie with id: " . $id;
15 });
16
17 $app->delete('/movies/{id}', function($id){
18     return "Will delete the movie with id: " . $id;
19 });
20
```

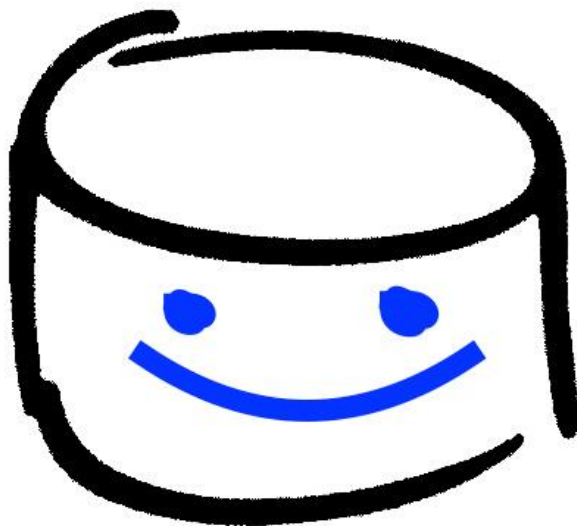
```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class MoviesController extends Controller
8 {
9
10     public function index(){
11         $movies = json_decode(file_get_contents('../resources/movies.json'));
12         return response()->json($movies);
13     }
14
15     public function show($id) {
16         $movies = json_decode(file_get_contents('../resources/movies.json'));
17
18         foreach ($movies->movies as $movie) {
19             if ($id == $movie->id) {
20                 return response()->json($movie);
21             }
22         }
23
24         return "No movie found";
25     }
26 }
```

En controllers uppgift

- Sköter logiken för vår applikation
- Tar emot anropet
 - Och ev. användardata
- Hanterar anropet
 - T.ex. hämta / skriva lämplig data i vår tjänst
 - Databas (MySQL, etc.)
 - Icke-relationella databaser
- Returnera ett svar
 - HTML-svar för en webbplats
 - JSON-svar för vårt API



Hmm. Hämta information från en
databas verkar ju najs...



Vi behöver en databas!



Våra filmer borde leva vidare!

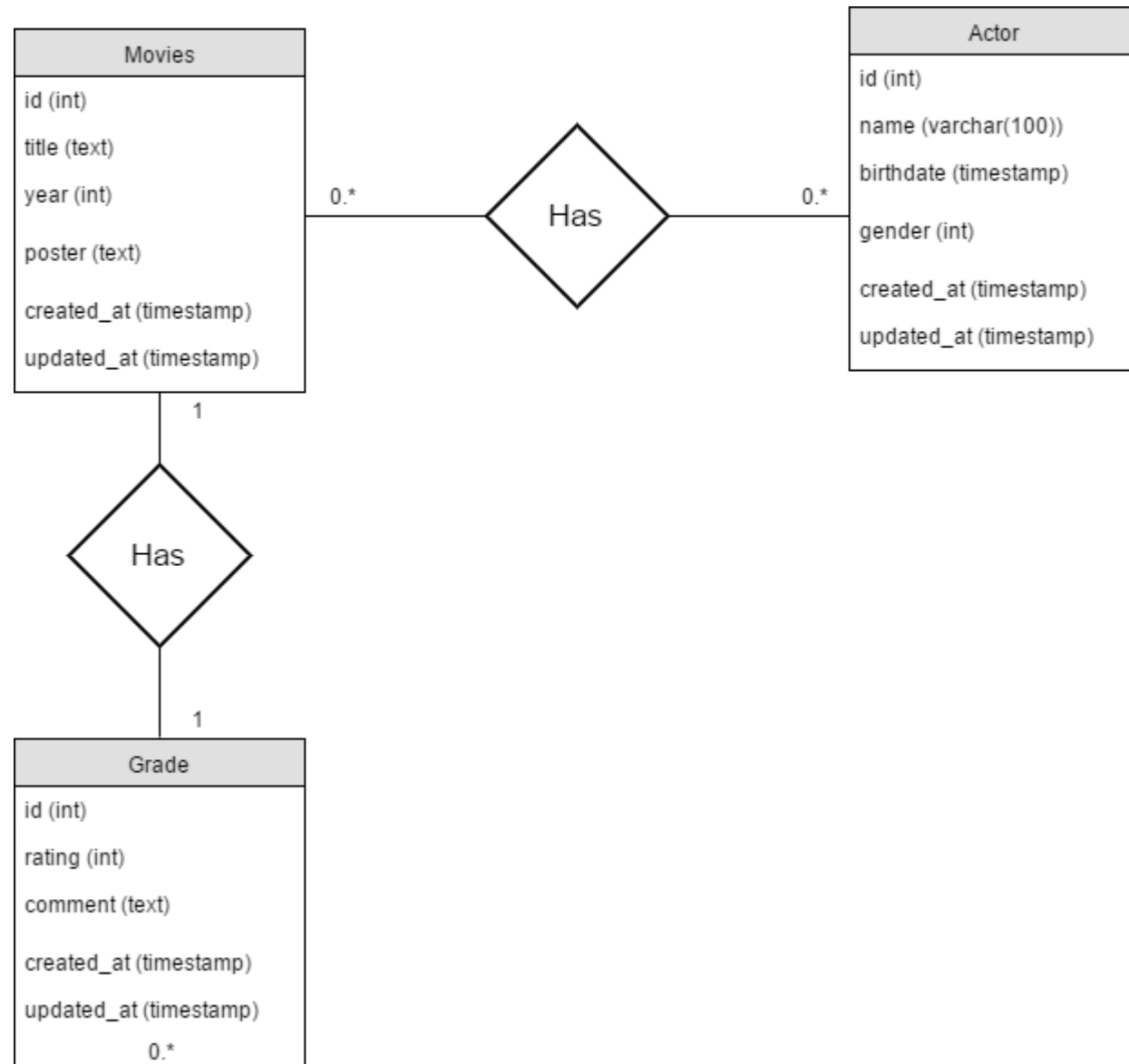
Vi vill att vårt API ska ha följande resurser

- Filmer
 - Lista alla filmer
 - Lista specifik film
 - Lägga till en film
 - Radera en film
 - Uppdatera en film
- Skådespelare
 - ...
- Betyg
 - ...



Rita ER-diagram!





Versionshantering av databas

- Migrations

Mycket smidigt när man är många i projektet!

- Hur hanterar ni databaser idag?

Varför an

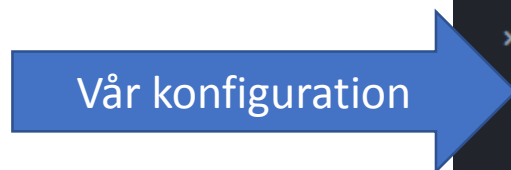
Ok, NOW we can panic!

- "Har du den
- "Vilken tabel
- "Ändrade du
- "Jäklar, jag tr
- "Inget funge
- berätta för m
- "Nä, nu vill ja



an att

Konfigurera vår databas: .env



Vår konfiguration

```
Program Platser System
.env — ~/Skrivbord/Labs/db-test — Atom
File Edit View Selection Find Packages Help

db-test
├── app
├── bootstrap
├── database
├── public
├── resources
├── routes
├── storage
├── tests
├── vendor
├── .env
├── .env.example
├── .gitignore
├── artisan
├── composer.json
├── composer.lock
├── phpunit.xml
└── readme.md
```

```
1 APP_ENV=local
2 APP_DEBUG=true
3 APP_KEY=
4 APP_TIMEZONE=UTC
5
6 DB_CONNECTION=mysql
7 DB_HOST=127.0.0.1
8 DB_PORT=3306
9 DB_DATABASE=movies
10 DB_USERNAME=root
11 DB_PASSWORD=supersecret
12
13 CACHE_DRIVER=memcached
14 QUEUE_DRIVER=sync
15
```

php artisan

Ett hjälpmedel för att snabba upp utvecklingen av webbapplikationer

Skapa en migration

Migration-funktioner

```
student@php-dev: ~/Skrivbord/Labs/db-test
Arkiv Redigera Visa Sök Terminal Hjälp

Available commands:
help           Displays help for a command
list           Lists commands
migrate        Run the database migrations
auth
auth:clear-resets  Flush expired password reset tokens
cache
cache:clear      Flush the application cache
cache:forget     Remove an item from the cache
cache:table      Create a migration for the cache database table
db
db:seed         Seed the database with records
make
make:migration   Create a new migration file
make:seeder      Create a new seeder class
migrate
migrate:install  Create the migration repository
migrate:refresh  Reset and re-run all migrations
migrate:reset    Rollback all database migrations
migrate:rollback Rollback the last database migration
migrate:status   Show the status of each migration
queue
queue:failed     List all of the failed queue jobs
queue:failed-table Create a migration for the failed queue jobs database table
queue:flush      Flush all of the failed queue jobs
queue:forget     Delete a failed queue job
queue:listen     Listen to a given queue
queue:restart    Restart queue worker daemons after their current job
queue:retry      Retry a failed queue job
queue:table      Create a migration for the queue jobs database table
queue:work       Start processing jobs on the queue as a daemon
schedule
schedule:run     Run the scheduled commands
student@php-dev:~/Skrivbord/Labs/db-test$ php artisan
```

```
php artisan make:migration create_movies_table
```

Vi skapar en migration för att skapa tabellen "movies"

Vår migration-fil

```
> app
> bootstrap
▼ database
  > factories
  ▼ migrations
    .gitkeep
    2017_04_18_154231_create_movies_table.php
  > seeds
> public
> resources
> routes
> storage
> tests
> vendor
.env
.env.example
.gitignore
artisan
composer.json
composer.lock
phpunit.xml
readme.md
```

Vår migration-fil

```
> app
> bootstrap
▼ database
  > factories
  ▼ migrations
    .gitkeep
    2017_04_18_154231_create_movies_table.php
  > seeds
> public
> resources
> routes
> storage
> tests
> vendor
.env
.env.example
.gitignore
artisan
composer.json
composer.lock
phpunit.xml
readme.md
```

```
1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateMoviesTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('movies', function (Blueprint $table) {
17             $table->increments('id');
18             $table->text('title');
19             $table->integer('year');
20             $table->text('poster');
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::drop('movies');
33     }
34 }
```


php artisan migrate

Verkställa våra migrations

Program Platser System

localhost / localhost / movies / movies | phpMyAdmin 4.5.4.1deb2ubuntu2 - Mozilla Firefox

php - Laravel/Lume...localhost / localhos...http://loca...80/movies/1Json Parser Online

localhost/phpmyadmin/tbl_structure.php?db=movies&table=movies&token=e8a0aa095d139ad25073f2ba12f9125d

Search

Star, Download, Home, etc.

phpMyAdmin

Recent Favorites

New

- information_schema
- movies
- mysql
- performance_schema
- phpmyadmin
- sys

Server: localhost » Database: movies » Table: movies

Browse

Structure

SQL

Search

Insert

Export

Import

Privileges

Operations

Tracking

Triggers

Table structure

Relation view

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	1 id	int(10)		UNSIGNED	No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial Fulltext Distinct values More
<input type="checkbox"/>	2 title	text	utf8_unicode_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
<input type="checkbox"/>	3 year	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
<input type="checkbox"/>	4 poster	text	utf8_unicode_ci		No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
<input type="checkbox"/>	5 created_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values More
<input type="checkbox"/>	6 updated_at	timestamp			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values More

☐ Check all

With selected:

Browse Change Drop Primary Unique Index Add to central columns Remove from central columns

Print view Propose table structure Track table Move columns Improve table structure

Add

1

column(s)

after updated_at

Go

+ Indexes

Information

Space usage

Data	16 KiB
Index	0 B

Row statistics

Format	dynamic
Collation	utf8_unicode_ci

```
php artisan make:migration create_actors_table  
php artisan make:migration create_grades_table  
php artisan make:migration create_grade_movie_table
```

Vi skapar en migration för att skapa tabellerna "actors", "grades", "grade_movie"

... och vi bygger klart resten av
databasen!

php artisan migrate

Verkställa våra migrations



Varför var detta nu bra då? 😊

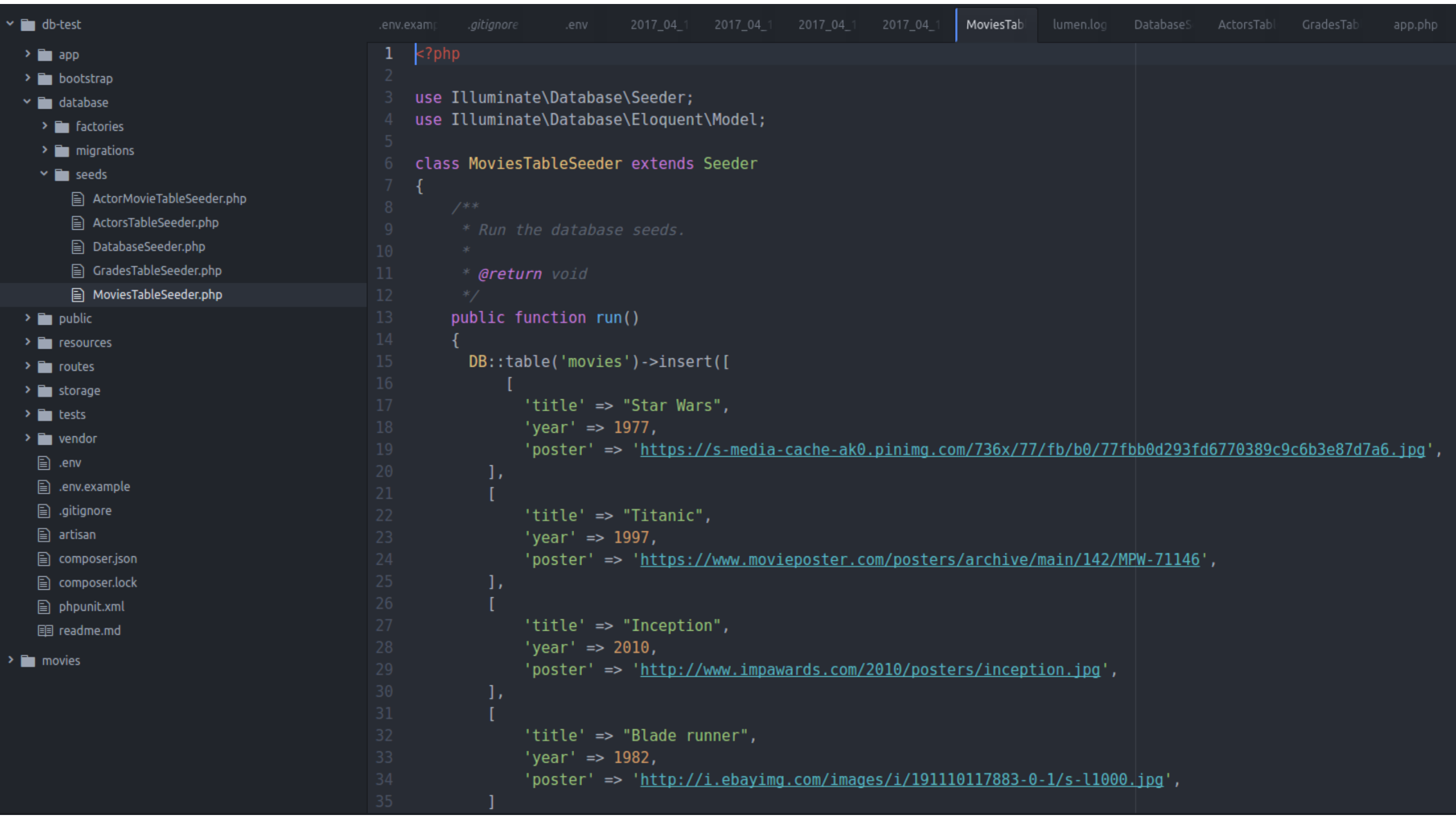


Hmm. Trist att databasen är helt tum 😞

Seeds! Lägg in demodata i
databasen!


```
php artisan make:seeder MoviesTableSeeder
```

Skapa en seed-fil för våra filmer

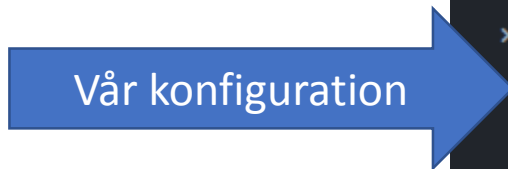


Nu har vi en databas, nu vill vi
göra något kul!

Lista alla filmer, etc.

Lumen – Kommunnicera med databaser

Konfigurera vår databas: .env



Vår konfiguration

```
Program Platser System
.env — ~/Skrivbord/Labs/db-test — Atom
File Edit View Selection Find Packages Help

db-test
├── app
├── bootstrap
├── database
├── public
├── resources
├── routes
├── storage
├── tests
├── vendor
├── .env
├── .env.example
├── .gitignore
├── artisan
├── composer.json
├── composer.lock
├── phpunit.xml
└── readme.md
```

```
1 APP_ENV=local
2 APP_DEBUG=true
3 APP_KEY=
4 APP_TIMEZONE=UTC
5
6 DB_CONNECTION=mysql
7 DB_HOST=127.0.0.1
8 DB_PORT=3306
9 DB_DATABASE=movies
10 DB_USERNAME=root
11 DB_PASSWORD=supersecret
12
13 CACHE_DRIVER=memcached
14 QUEUE_DRIVER=sync
15
```

Basic Usage

Note: If you would like to use the `DB` facade, you should uncomment the `$app->withFacades()` call in your `bootstrap/app.php` file.

For example, without facades enabled, you may access a database connection via the `app` helper:

```
$results = app('db')->select("SELECT * FROM users");
```

Or, with facades enabled, you may access the database connection via the `DB` facade:

```
$results = DB::select("SELECT * FROM users");
```

Basic Queries

To learn how to execute basic, raw SQL queries via the database component, you may consult the [full Laravel documentation](#).

Query Builder

Lumen may also utilize the Laravel fluent query builder. To learn more about this feature, consult the [full Laravel documentation](#).

Eloquent ORM

If you would like to use the Eloquent ORM, you should uncomment the `$app->withEloquent()` call in your `bootstrap/app.php` file.

Of course, you may easily use the full Eloquent ORM with Lumen. To learn how to use Eloquent, check out the [full Laravel documentation](#).

Aktivara "Facades"



```
.env 2017_04_1 2017_04_1 2017_04_1 2017_04_1
10
11 /*
12 |-----|
13 | Create The Application
14 |-----|
15 |
16 | Here we will load the environment and
17 | that serves as the central piece of th
18 | application as an "IoC" container and
19 |
20 */
21
22 $app = new Laravel\Lumen\Application(
23     realpath(__DIR__.'/../')
24 );
25
26 $app->withFacades();
27 $app->withEloquent();
28
29 /*
```

Avkommentera
Dessa rader

Vi testar! 😊

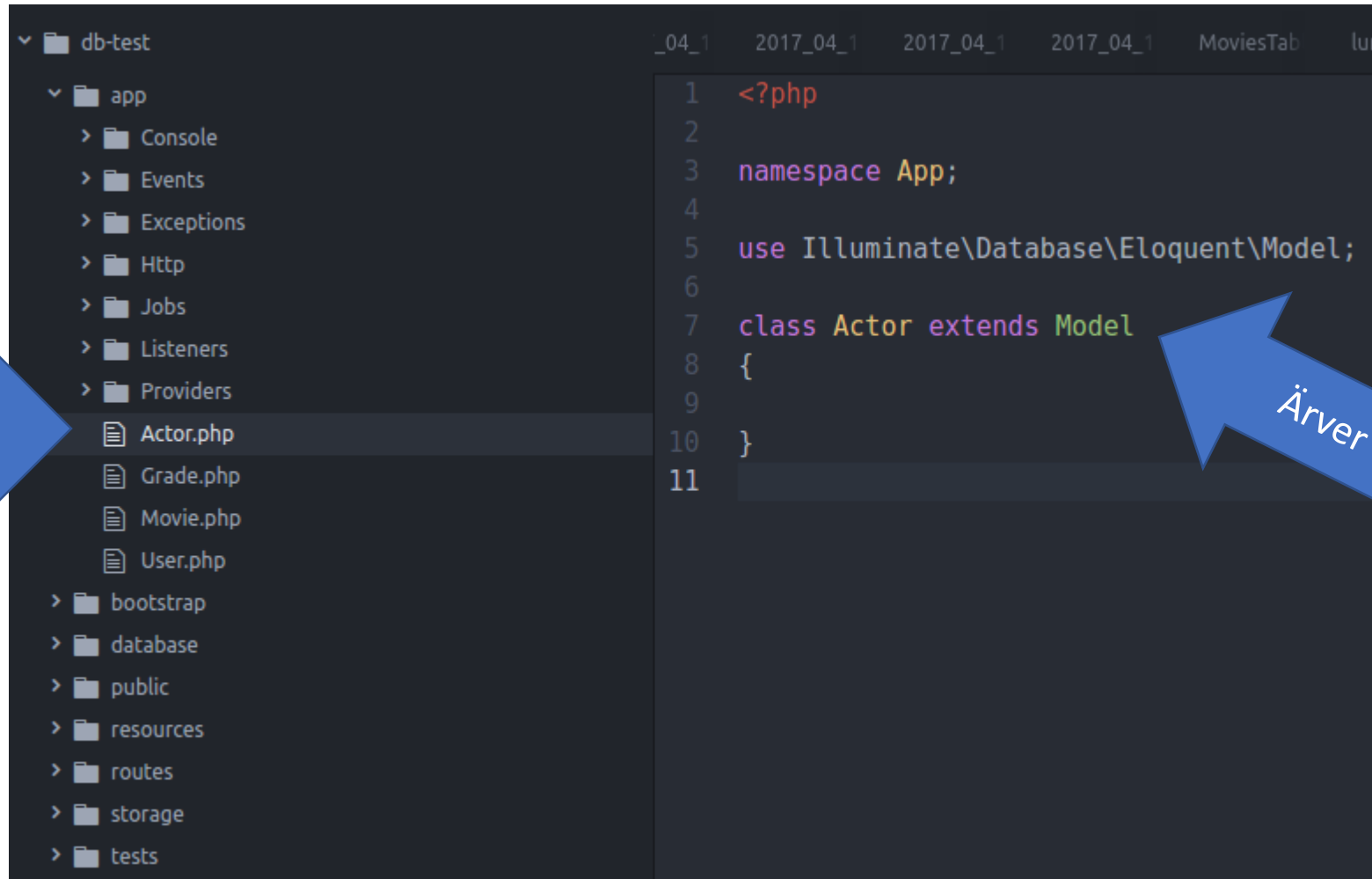
Models & ORM

*”ORM, står för **object-relational mapping**, är ett objektorienterat system som konverterar databastabeller till klasser, tabellrader till objekt, samt celler till objekt-attribut”*

En model är en...

- Representation av en resurs
- I Lumen så är dessa väldigt ofta (men inte alltid) kopplade till en databastabell
- Alltså, eftersom vi har 3st tabeller kommer vi skapa 3st modeller
 - Movies
 - Actors
 - Grades
- Vi har även en många till många-relation som är en egen tabell & således en egen modell (om vi vill!)
 - ActorMovie

Var finns modellerna?



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

Ärver av Model

Bra att veta om modeller

- Mappas mot en tabell i databasen, fast i plural. T.ex.
 - Modellen "Movie" mappas mot tabellen "movies" i databasen
 - Kan ändras genom egenskapen (i modellen):
 - `protected $table = 'my_movies';`
- Andra bra egenskaper kan ni läsa mer om här:
<https://laravel.com/docs/5.4/eloquent> , t.ex.
 - Vilka värden som får fyllas i
 - Vilken databasanslutning som gäller (om man har flera)
 - Om man ska använda "softDelete", m.m.

Vi testar! 😊