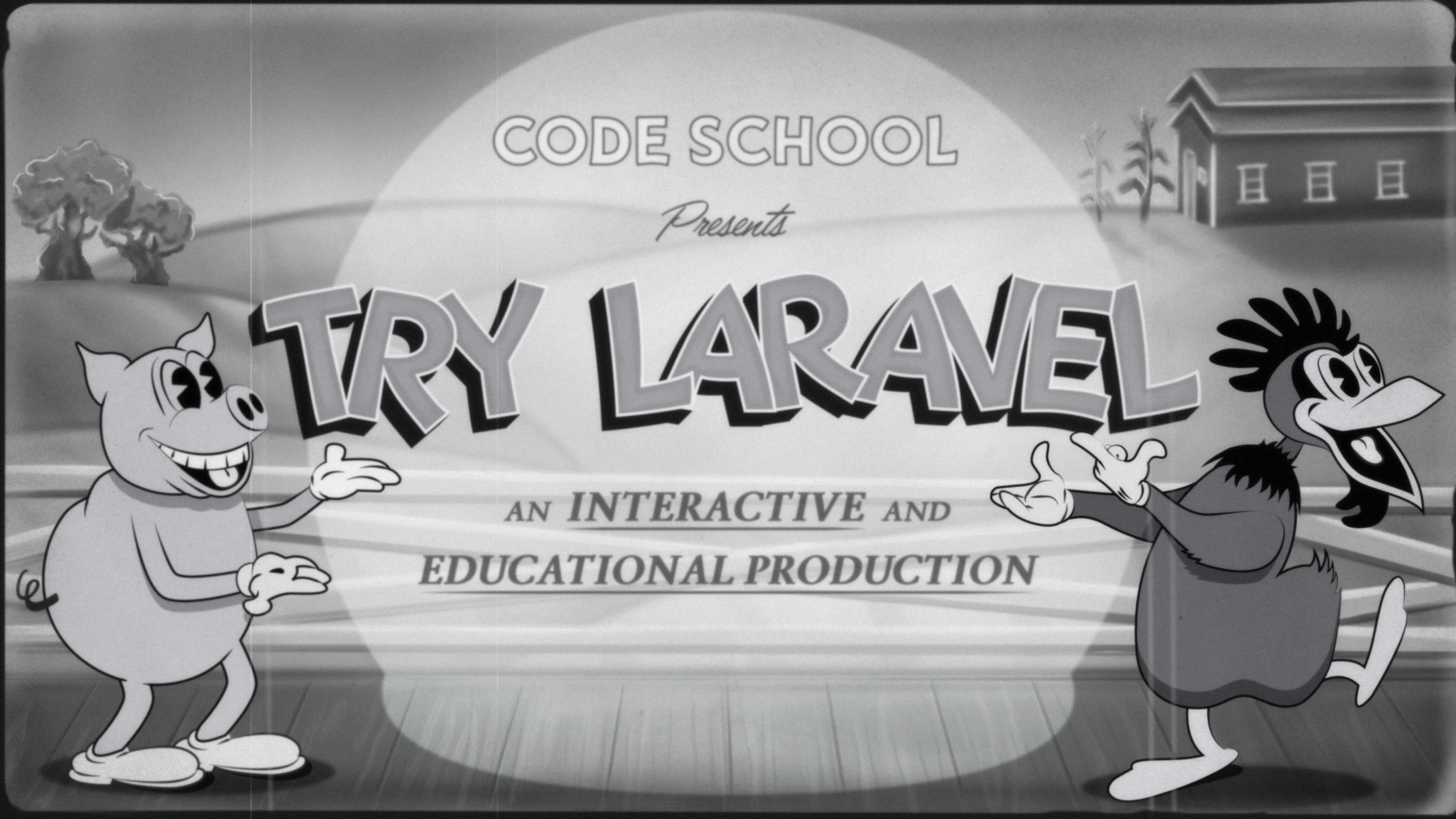

CODE SCHOOL
Presents
TRY LARAVEL
AN INTERACTIVE AND
EDUCATIONAL PRODUCTION

Level 1 – Section 1

Welcome to Laravel

What Are We Doing Here?

Prerequisites of Try Laravel

Basic & Some Object-oriented PHP

Try PHP & Close Encounters With PHP

Basic Database Knowledge

Try SQL

TRY LARAVEL

What Is Laravel?

Laravel is a web framework built in PHP.

Open source!

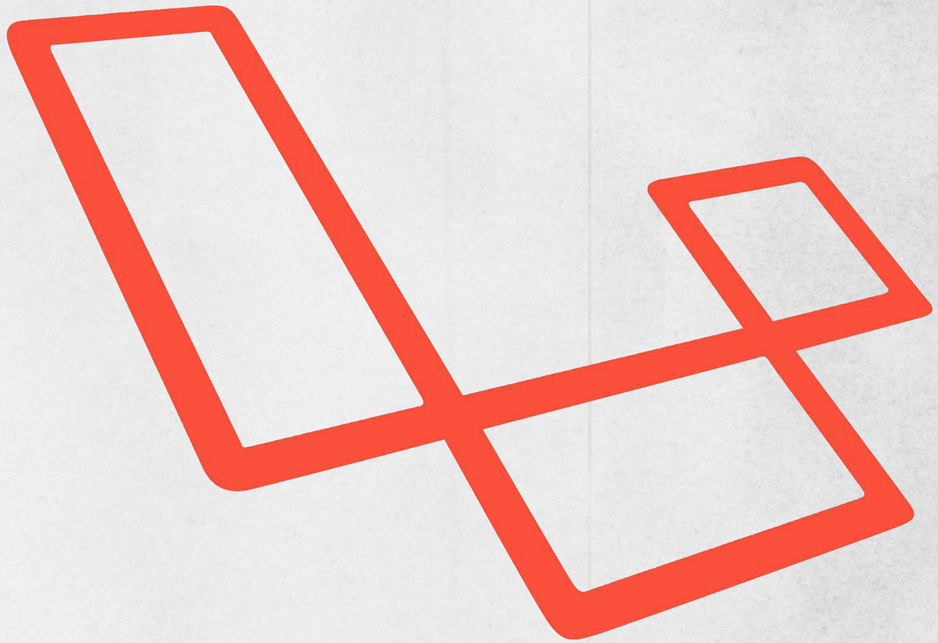
Initial release in 2011, currently in Version 5.4.

Built for developing applications using the MVC (model view controller) pattern.

Uses RESTful controllers for predictable URL patterns.

Built-in database ORM and migrations.

Far too many features to list!



TRY LARAVEL

Local Farmers Market Listing

Markets

Orlando Market

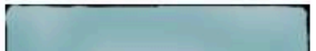
AP Community Market

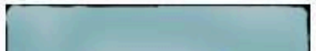
Lake Lilly Market

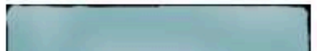
Cair Paravel Market

Heidenreich and Sons

Rath-Schneider







Creating a New Farmers Market

CREATE A MARKET

Market Name

Your Market's Name

Market City

Market Location City

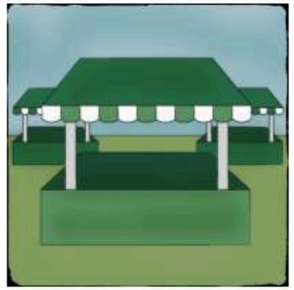
Market Website

Market Website

Create

Viewing Each Market’s Details

Lake Lilly Market



FARMS AT THIS MARKET

- Bailey-Hackett
- Lueilwitz Inc
- Murphy-Toy
- Mertz-O'Keefe
- Wehner, Gleichner and Cassin

Location City: Maitland

Website:
lakelillymarket.com

Editing Existing Markets

EDIT

Lake Lilly Market

Market Name

Lake Lilly Market

Market City

Maitland

Market Website

lakelillymarket.com

Update

Databases, Like a Spreadsheet

Here's a listing of farmers markets in a spreadsheet form.

Columns

markets table

Rows

	A	B	C
1	Orlando Market	orlandofarmersmarket.com	Orlando
2	AP Community Market	apcommunitymarket.com	Orlando
3	Lake Lilly Market	lakelillymarket.com	Maitland
4	Cair Paravel Market	cairparavelmarket.com	Narnia

id *name* *website* *city*

Collections, Data Organized

The collect function is used to create a collection from an array.

```
$c = collect(['Orlando Farmers Market', 'AP Community Market']);
```

```
$c = collect([
                ['id' => 1,
                 'name' => 'Orlando Farmers Market',
                 'city' => 'Orlando'],
                ['id' => 2,
                 'name' => 'AP Community Market',
                 'city' => 'Orlando'],
                ['id' => 3,
                 'name' => 'Lake Lilly Market',
                 'city' => 'Maitland'],
             ]);
```

New column

New row

Accessing Collection Data

There are many useful methods we can use with a collection.

```
$c = collect(['Orlando Farmers Market', 'AP Community Market']);
```

```
$c->first();
```

=> "Orlando Farmers Market"

```
$c->last();
```

=> "AP Community Market"

Accessing Larger Collection Data

Many methods allow us to get specific data from each item in the collection.

```
$c = collect([
                ['id' => 1,
                 'name' => 'Orlando Farmers Market',
                 'city' => 'Orlando'],
                ['id' => 2,
                 'name' => 'AP Community Market',
                 'city' => 'Orlando'],
                ['id' => 3,
                 'name' => 'Lake Lilly Market',
                 'city' => 'Maitland'],
              ]);
```

```
$c->fetch('name');
```

=> ["Orlando Farmers Market", "AP Community Market" , "Lake Lilly Market"]

Retrieving a Collection From the Database

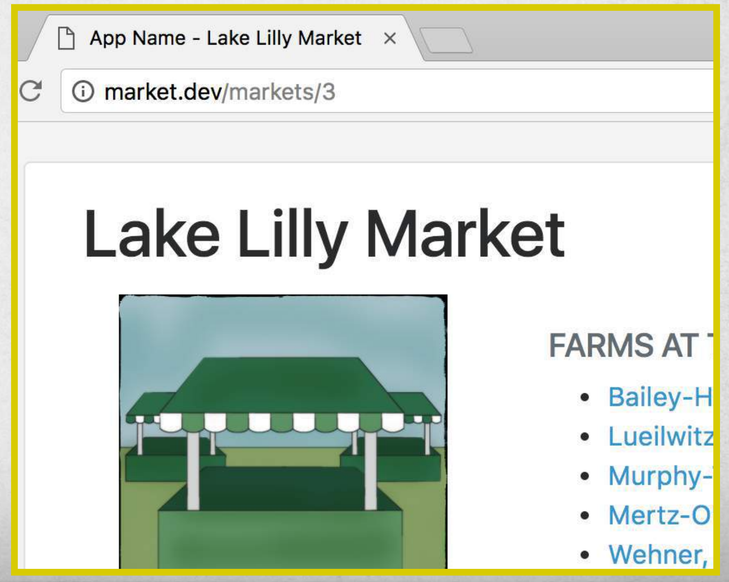
How can we get the market with an id of 3.

markets

id	name	website	city
1	Orlando Market	orlandofarmersmarket.com	Orlando
2	AP Community Market	apcommunitymarket.com	Orlando
3	Lake Lilly Market	lakelillymarket.com	Maitland
4	Cair Paravel Market	cairparavelmarket.com	Narnia

Desired result

```
=> {['id' => 3,
     'name' => 'Lake Lilly Market',
     'city' => 'Maitland'
   ]}
```



Using Eloquent to Find in the Database

Accessing collection data through Eloquent with find.

```
$m = Market::find(3);
```

```
=> App\Market{
     'id': 3, 'name': 'Lake Lilly Market','city': 'Maitland'
   }
```

The class name Market links to our table name "markets"

```
$m['name'];
```

```
=> "Lake Lilly Market"
```

```
$m->name;
```

```
=> "Lake Lilly Market"
```

Laravel in Review

What is Laravel?

Laravel's collection class

Reading from the database with Eloquent

Level 1 – Section 2

Welcome to Laravel

Cleaning Up CRUD

Creating a Market

Using our model to create a new market

Create a Market

```
// single attribute method
$m = new Market;

$m->name = 'Winter Garden Market';

$m->save();
```

Create a new Market object

Set the name property value

save the new market to the database

Mass Assignment Creation

We can also pass multiple values in an array to create a new item.

```
// mass assignment
$data = ['name' => 'Winter Garden Market', 'city' => 'Winter Garden'];

Market::create($data);
```

Create an array of properties and values

Create and save a new market using the $data array

Reading From a Market

Here's how we can use our model to read data from a market.

```
// single item
$m = Market::find(3);

// all items, in a collection
$markets = Market::all();

// custom constraint
$markets = Market::where('city', 'Orlando')->get();
```

Find the market with the id of 3

Find all markets

Find all markets where the city is Orlando

Reading Markets With Custom Constraints

Custom methods can be chained together to make a more powerful query.

```
// multiple constraints
$markets = Market::where('city', 'Orlando')
                ->orderBy('name', 'desc')
                ->take(5)
                ->get();
```

Find markets in Orlando only

Order by the market name, in descending order

Only the first 5

Run the query


```
[
  - {
        id: 5,
        name: "Pine Hills Market",
        city: "Orlando",
        website: "phmarket.com",
        created_at: "2017-01-06 13:30:50",
        updated_at: "2017-02-03 02:05:27"
    },
  - {
        id: 1,
```

Updating a Market

Here's how we can use our model to update data of a market.

```
// single item update
$m = Market::find(3);

$m->name = 'Winter Garden Co-Op Market';

$m->save();
```

Find a single market

Change the name property

Update the market in the database

Mass Updates

We can also update multiple attributes at the same time using an array of key-value pairs.

```
// mass update
$m = Market::find(3);

$data = [
          'name' => 'Winter Gar                    arket',
          'website' => 'wgcoop.com
        ];

$m->fill($data);
```

new title

Create an array of values to change

Use fill to update with the new data

Deleting One or More Markets

Here's how we can use our model to delete a market.

```
// single record deletion
$m = Market::find(3);

$m->delete();

// single record destroy
Market::destroy(3);

// multiple record destroy
Market::destroy([3, 4, 5]);
```

Find a market by id

Delete found market

Destroy a single or multiple markets by id

The Power of Eloquent

Using our model to perform CRUD actions in Eloquent is simple.

Create a Market

```
$m = new Market;

$m->name = 'Winter Garden Market';

$m->save();
```

Read from a Market

```
$m = Market::find(3);

echo $m->name;

=> 'Lake Lilly Market'
```

Update a Market

```
$m = Market::find(3);

$m->name = 'Maitland Market';

$m->save();
```

Delete a Market

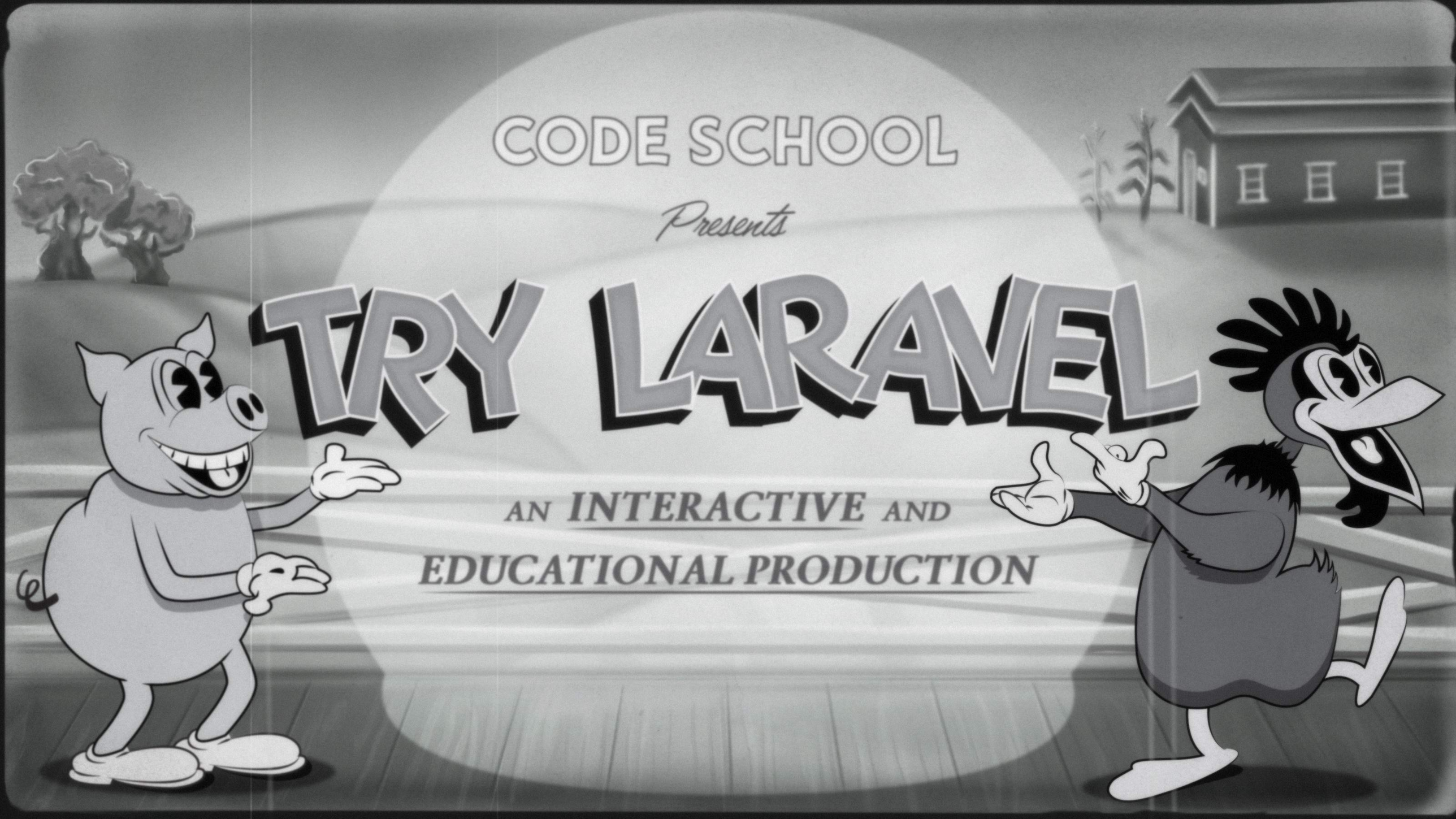
```
$m = Market::find(3);

$m->delete();
```

Laravel in Review

Laravel CRUD methods

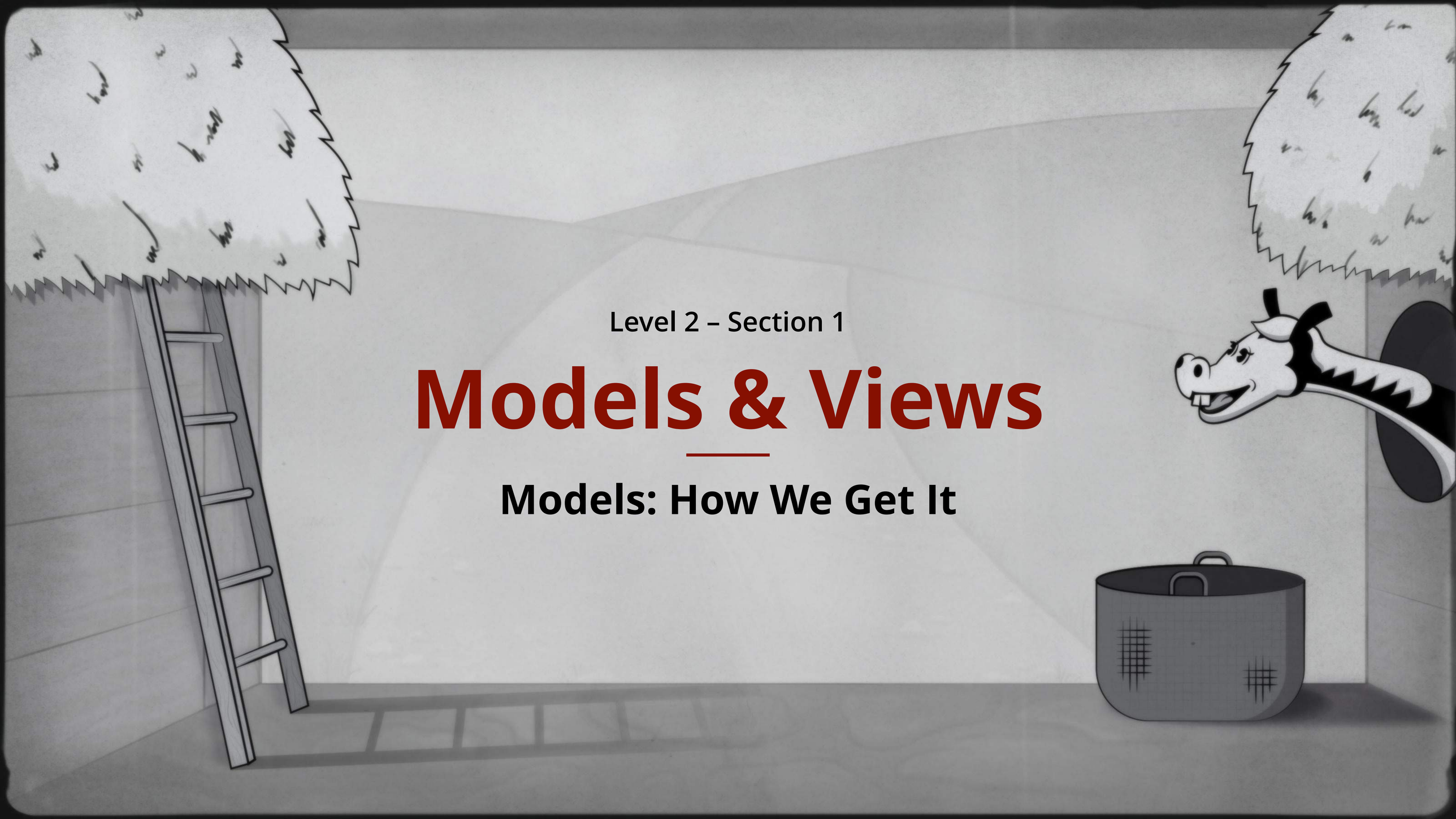
Mass assignment for create and update


CODE SCHOOL
Presents
TRY LARAVEL
AN INTERACTIVE AND
EDUCATIONAL PRODUCTION

Level 2 – Section 1

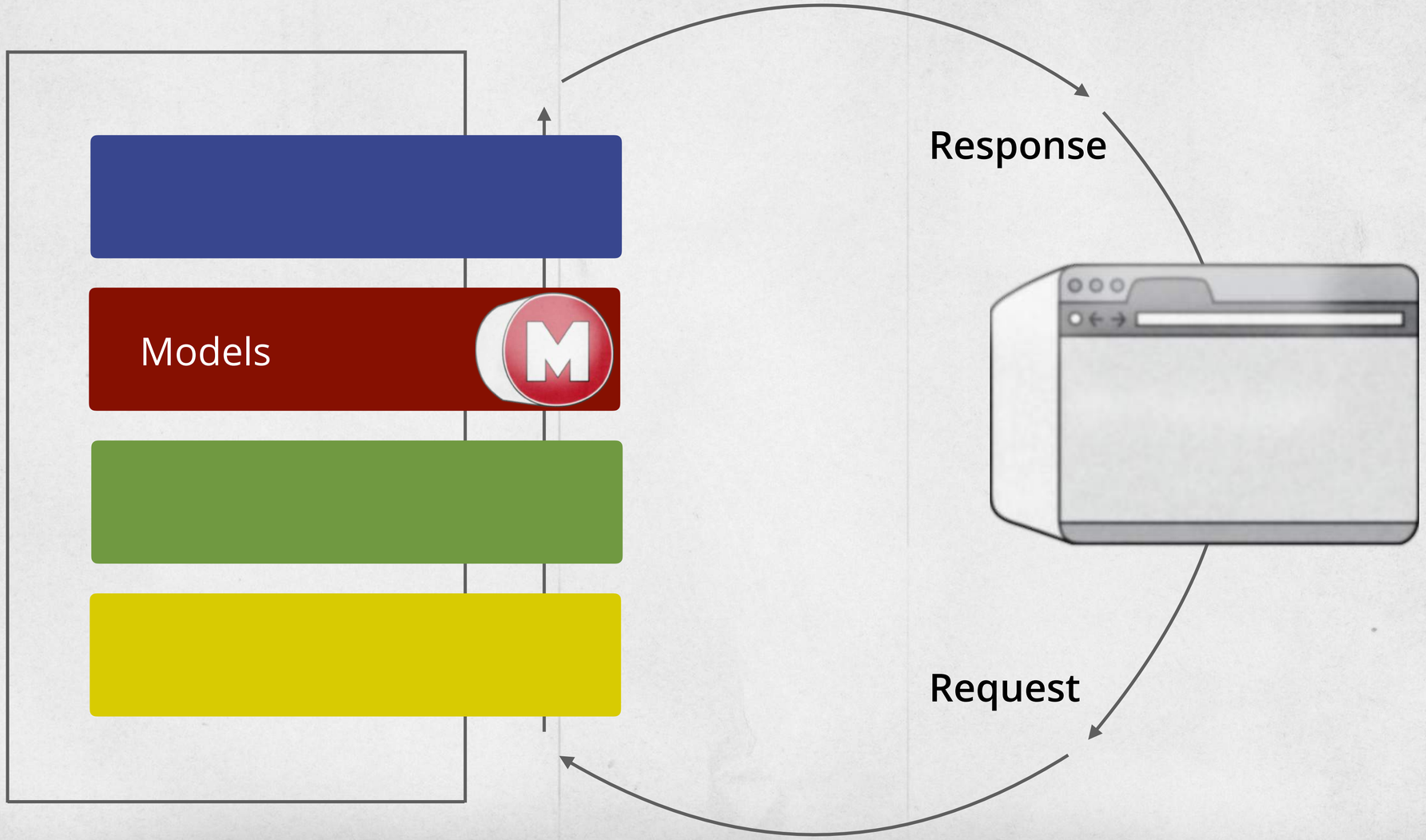
Models & Views

Models: How We Get It



Application Stack Diagram

The model is where we define our database interaction.



We Need a Collection From the Database

How can we get the market with an id of 3?

markets

id	name	website	city
1	Orlando Market	orlandofarmersmarket.com	Orlando
2	AP Community Market	apcommunitymarket.com	Orlando
3	Lake Lilly Market	lakelillymarket.com	Maitland
4	Cair Paravel Market	cairparavelmarket.com	Narnia

Desired result

```
=> {['id' => 3, 'name' => 'Lake Lilly Market','city' => 'Maitland']}
```

Eloquent Models Associate Database Tables

```
$m = Market::find(3);
```

app/Market.php

```
use Illuminate\Database\Eloquent\Model;

class Market extends Model
{
}
```

markets

id	name	website	city
1	Orlando Market	orlandofarmersmarket.com	Orlando
2	AP Community Market	apcommunitymarket.com	Orlando
3	Lake Lilly Market	lakelillymarket.com	Maitland
4	Cair Paravel Market	cairparavelmarket.com	Narnia

Eloquent Models Associate Database Tables

```
$m = Market::find(3);
```

app/Market.php

```
use Illuminate\Database\Eloquent\Model;

class Market extends Model
{
}
```

markets

id	name	website	city
1	Orlando Market	orlandofarmersmarket.com	Orlando
2	AP Community Market	apcommunitymarket.com	Orlando
3	Lake Lilly Market	lakelillymarket.com	Maitland
4	Cair Paravel Market	cairparavelmarket.com	Narnia

Managing Long Queries

What happens when you need to make longer queries more often?

```
// retrieve orlando markets
$orlandoMarkets = Market::where('city', 'Orlando')
                        ->orderBy('name', 'desc')
                        ->get();
```

How can we simplify this?

Local Scopes

Local scopes help you to create custom queries for common code.

```
class Market extends Model
{
    // only Orlando markets
    public function scopeOrlando($query)
    {
        return $query->where('city', 'Orlando')
                     ->orderBy('name', 'desc');
    }
}
```

The scope definition starts with the word scope followed by the name of the scope

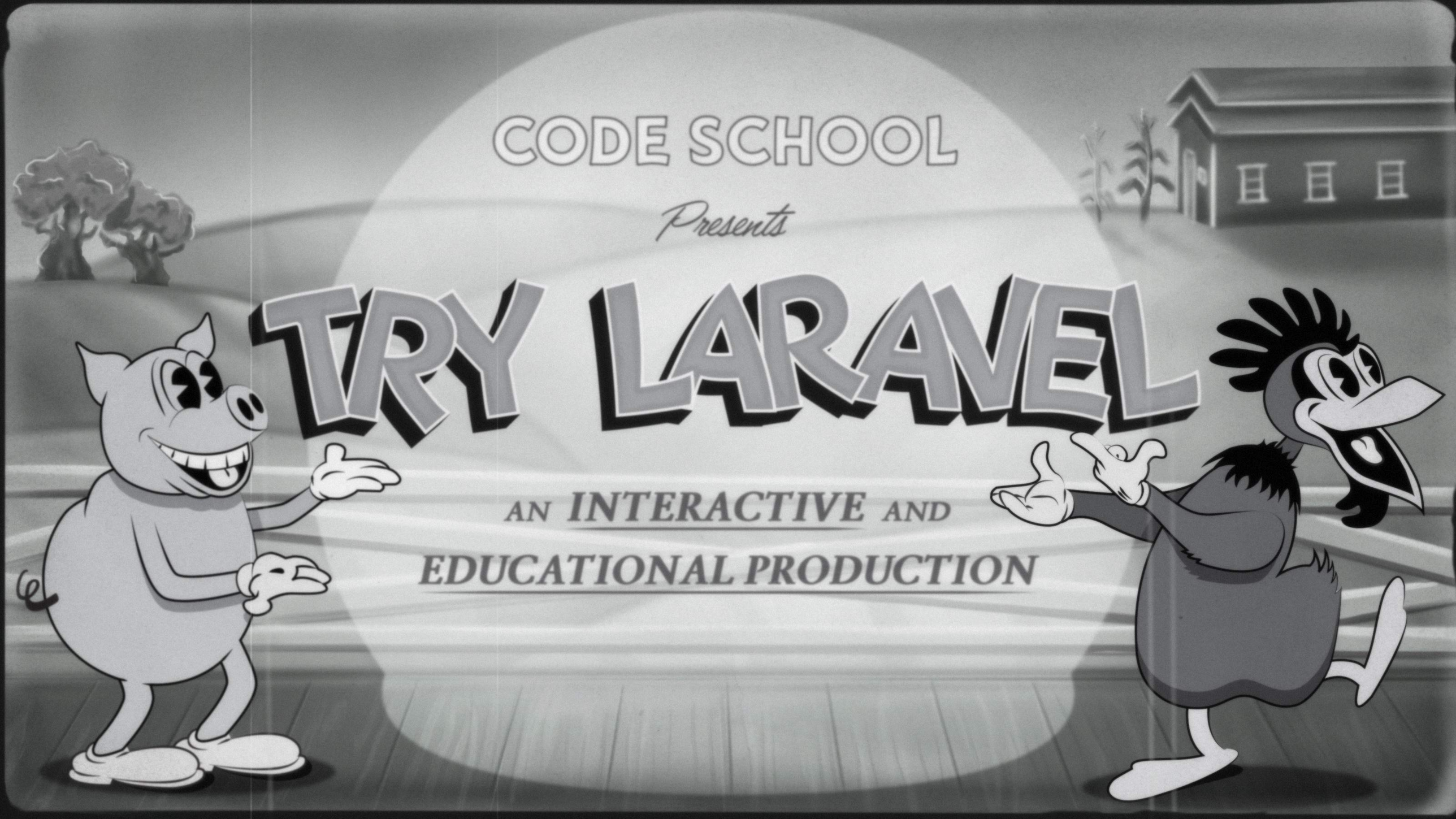
```
// retrieve orlando markets
$orlandoMarkets = Market::orlando()->get();
```

To call the scope, you just write the name without the word "scope" in front

Models in Review

Creating models with Laravel's Eloquent

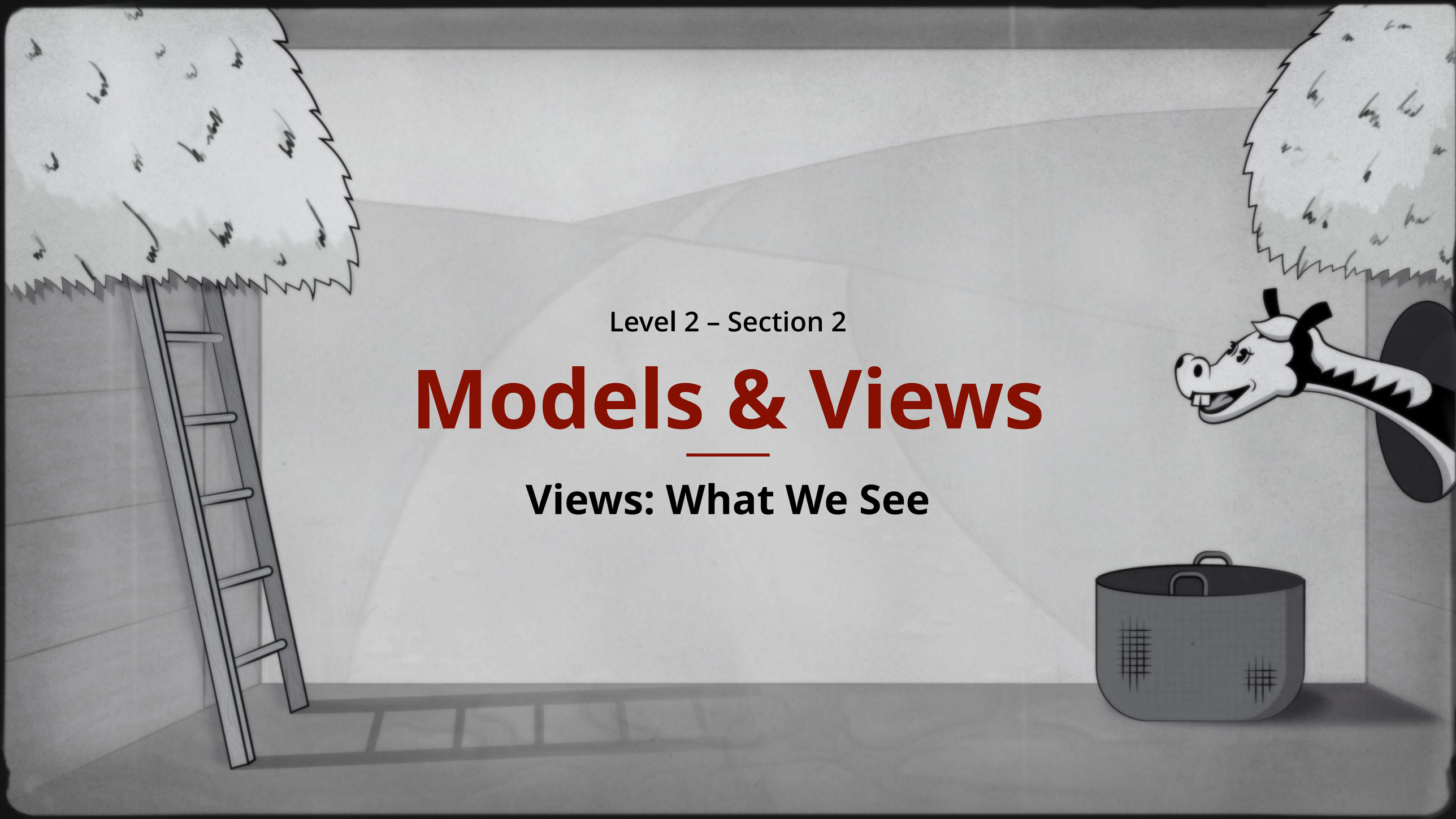
Local scopes inside of the model


CODE SCHOOL
Presents
TRY LARAVEL
AN INTERACTIVE AND
EDUCATIONAL PRODUCTION

Level 2 – Section 2

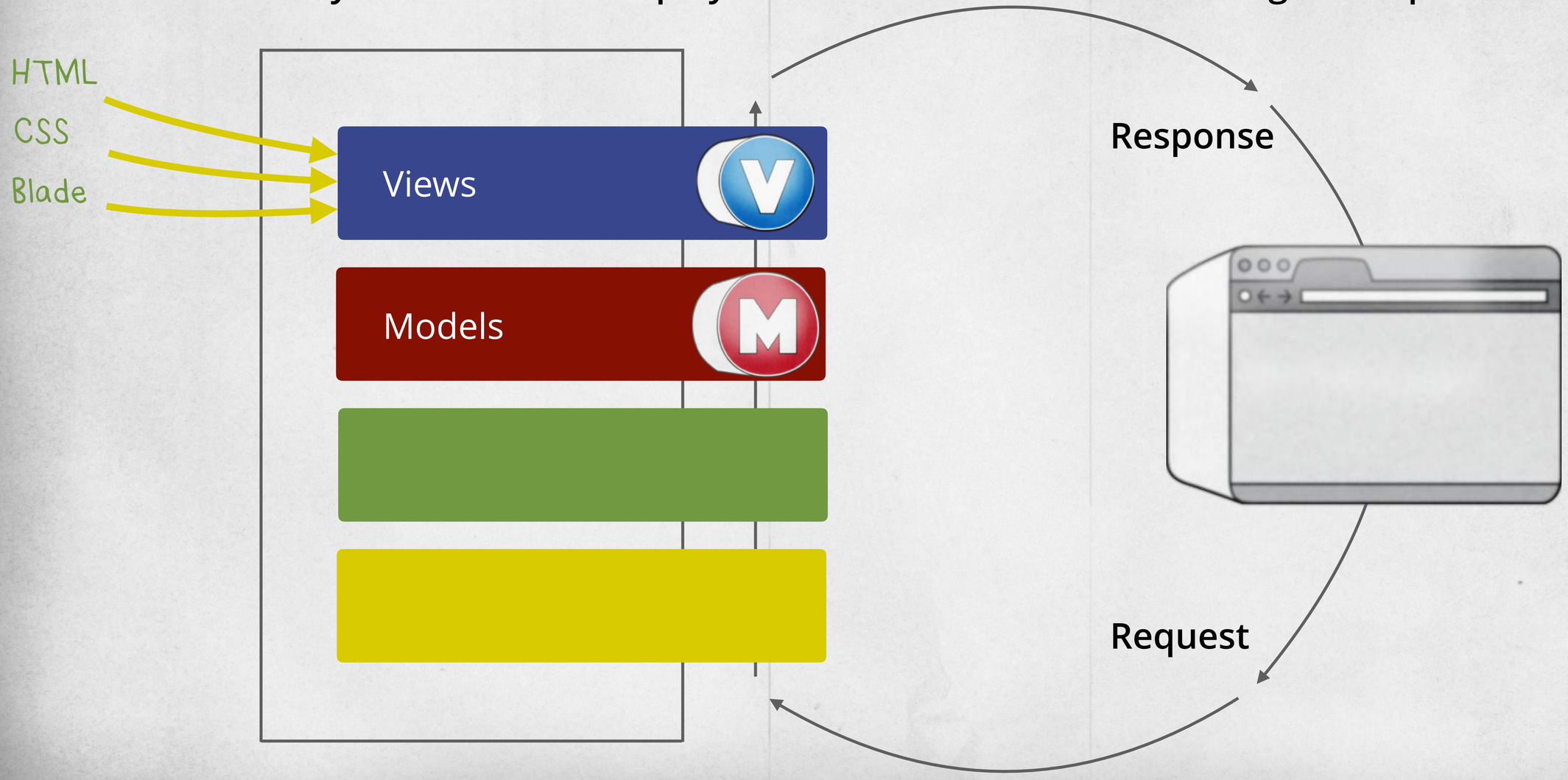
Models & Views

Views: What We See



Application Stack Diagram

The views layer is where we display our information back to the original request.



Our Project Files So Far

Laravel's folder structure is easy to understand and helps us organize our code.

Views live here, under the Resources/views folder

Any view specific to a model should live in their own folder under the views folder

Showing a Single Market With Blade

Resources/views/markets/show.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head><title>Farm To Market</title></head>
<body>
@php
  $market = Market::find(3);
@endphp

  <h1>{{ $market->name }}</h1>
  <h3>{{ $market->city }}</h3>
  Website:
  <a href="{{ $market->website }}">
    {{ $market->website }}
  </a>

</body>
</html>
```

Animations: Build HTML first, then @php section then the remainder

@php and @endphp behaves like <?php ?>

{{ }} echoes any code within!

View is rendered with no styles

Listing All Markets With a View

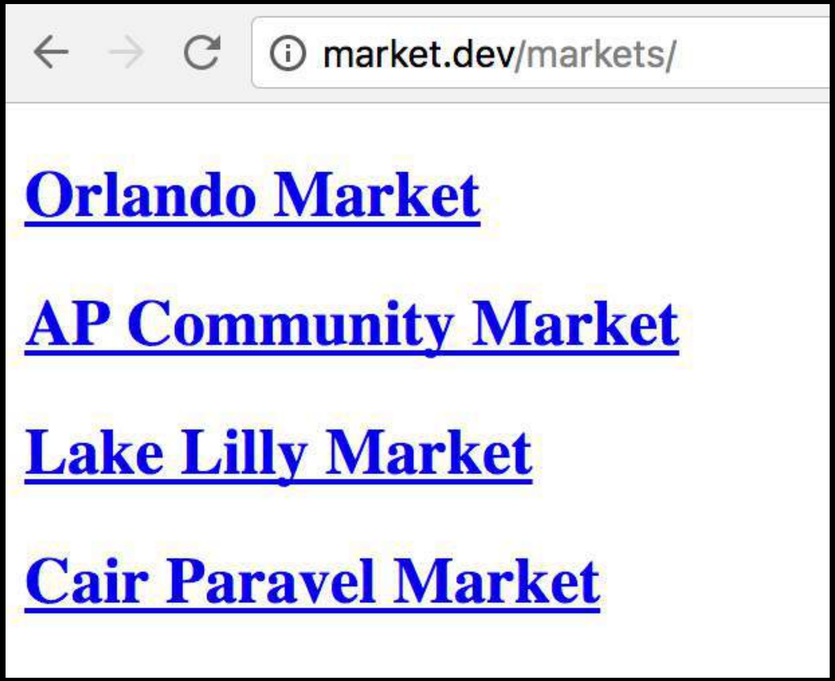
resources/views/markets/index.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head><title>Farm To Market</title></head>
<body>

@php
  $markets = Market::all();
@endphp

@foreach ($markets as $market)
  <a href="{{ $market->website }}">
    <h2>{{ $market->name }}</h2>
  </a>
@endforeach

</body>
</html>
```



Creating a Layout

resources/views/layouts/app.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head><title>Farm To Market</title><
<body>

  @yield('main')

</body>
</html>
```

resources/views/markets/index.blade.php

```
@extends('layouts.app')
@section('main')
  @php
    $markets = Market::all();
  @endphp
  @foreach ($markets as $market)
    <a href="{{ $market->website }}">
      <h2>{{ $market->name }}</h2>
    </a>
  @endforeach
@endsection
```

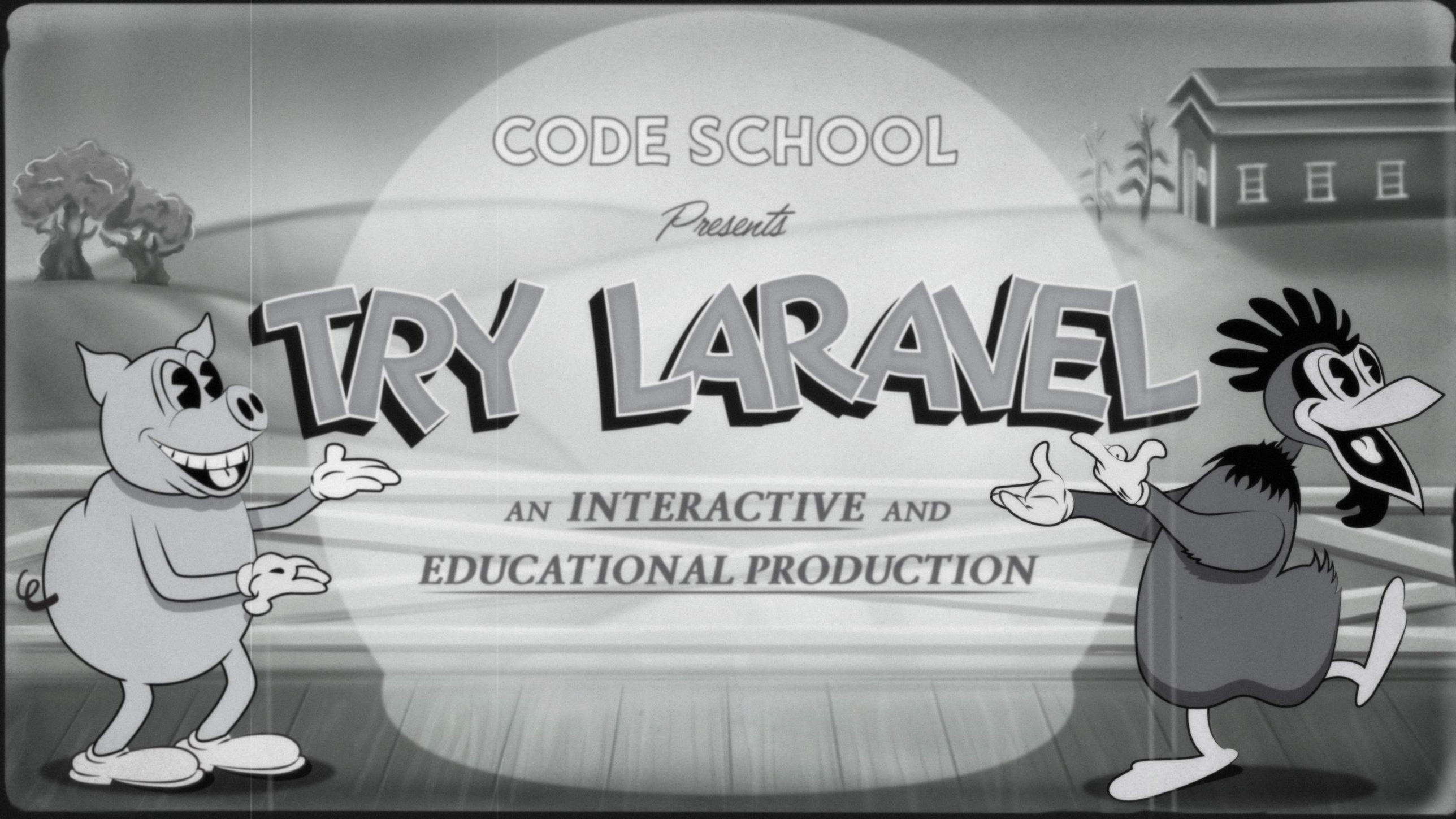
@extends starts by looking in the views folder!
In this example we are looking for:
views/layouts/app.blade.php

Views in Review

Creating views

Basic Blade template syntax

Using Blade to create layouts


CODE SCHOOL
Presents
TRY LARAVEL
AN INTERACTIVE AND
EDUCATIONAL PRODUCTION

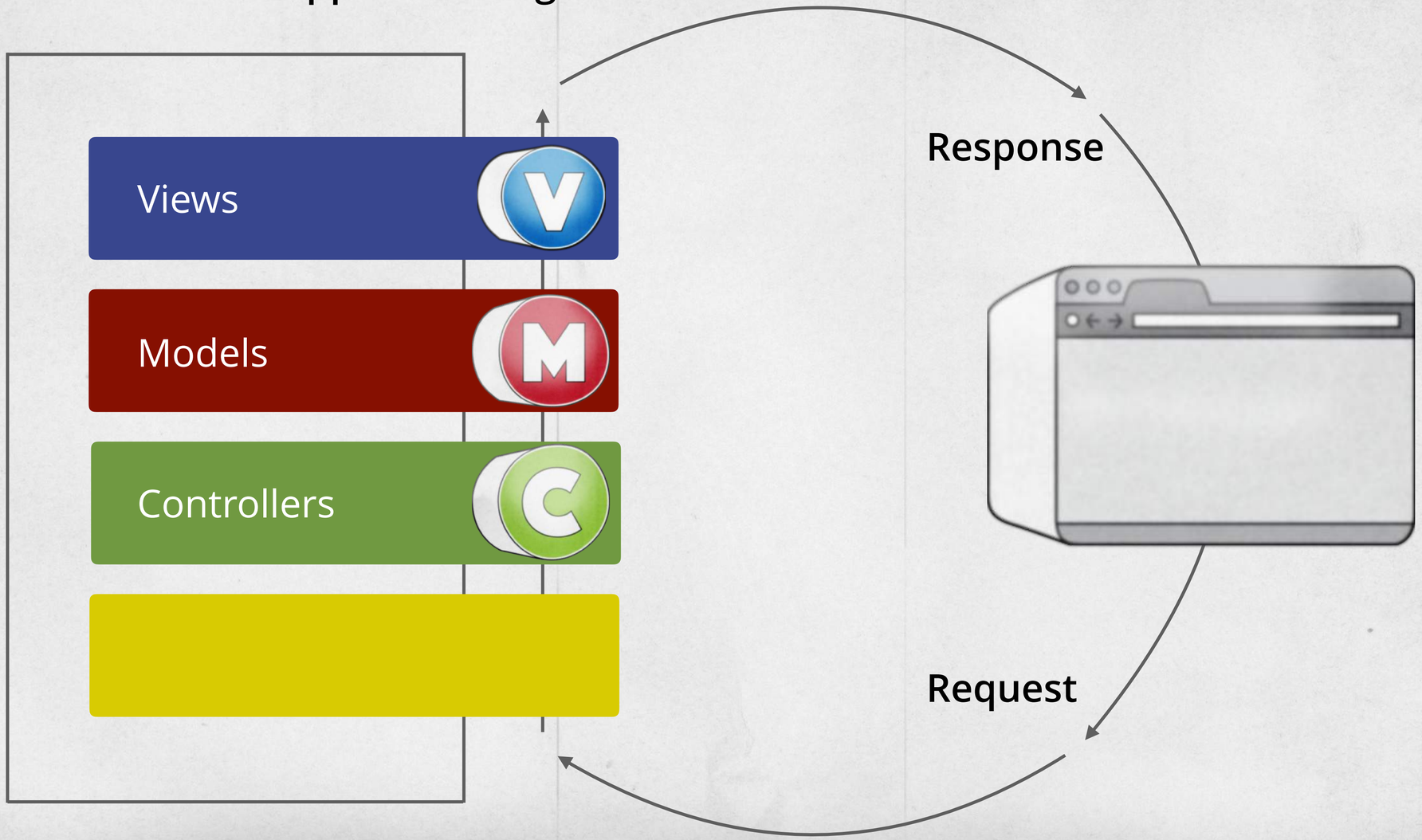
Level 3 – Section 1

Controllers & Routing

Controllers: The Middleman

Application Stack Diagram

Controllers handle the application logic and render our views.



Creating a New Controller

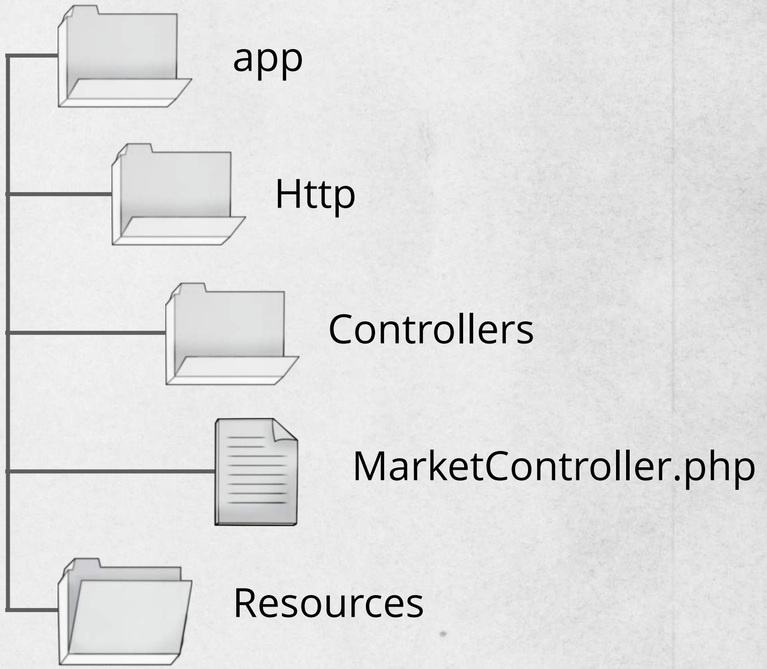
app/Http/Controllers/MarketController.php

```
namespace App\Http\Controllers;

use App\Market;
use Illuminate\Http\Request;

class MarketController extends Controller
{

}
```



Moving Logic to the Controller

app/Http/Controllers/MarketController.php

```
namespace App\Http\Controllers;

use App\Market;
use Illuminate\Http\Request;

class MarketController extends Controller
{
    // Show All Markets
    public function index()
    {

    }
}
```

This action will be used to view all markets

Moving Logic to the Controller

app/Http/Controllers/MarketController.php

```
namespace App\Http\Controllers;

use App\Market;
use Illuminate\Http\Request;

class MarketController extends Controller
{
    // Show All Markets
    public function index()
    {
        $markets = Market::all();
    }
}
```

Make a call to the Market model to get all markets!

Moving Logic to the Controller

app/Http/Controllers/MarketController.php

```
namespace App\Http\Controllers;

use App\Market;
use Illuminate\Http\Request;

class MarketController extends Controller
{
    // Show All Markets
    public function index()
    {
        $markets = Market::all();
        return view('markets.index', ['markets' => $markets]);
    }
}
```

The first argument is our view template location

The second argument is used to pass along any data we want to the view!

All Markets Using a Controller

Using the controller, we can now remove the call to query all markets.

Resources/views/markets/index.blade.php

```
@extends('layouts.app')
@section('main')

@php
  $markets = Market::all();
@endphp

@foreach ($markets as $market)
  <a href="{{ $market->site }}">
    <h2>{{ $market->name }}</h2>
  </a>
@endforeach
```

@php and @endphp in MVC
should be a sign of code smell

All Markets Using a Controller

Using the controller, we can now remove the call to query all markets.

Resources/views/markets/index.blade.php

```
@extends('layouts.app')
@section('main')

@php
  $markets = Market::all();
@endphp

@foreach ($markets as $market)
  <a href="{{ $market->site }}">
    <h2>{{ $market->name }}</h2>
  </a>
@endforeach
```

Markets are now passed from the controller!

Single Market Logic

To show a single market, we can create a new method in the controller called show.

app/Http/Controllers/MarketController.php

```
Class MarketController extends Controller
{
    // Show All Markets
    public function index()
    {
        $markets = Market::all();
        return view('markets.index', ['markets' => $markets]);
    }

    // Show Single Market
    public function show(Market $market)
    {
        return view('markets.show', ['market' => $market]);
    }
}
```

We are now passing an argument from the request to our method, the Market object

Showing a Single Market

Again, we can now just remove the logic that is in the controller.

Resources/views/markets/show.blade.php

```
@extends('layouts.app')
@section('main')

@php
  $market = Market::find(3);
@endphp

  <h1>{{ $market->name }}</h1>
  <h3>{{ $market->city }}</h3>
  site:
  <a href="{{ $market->site }}">
    {{ $market->site }}
  </a>
```

Showing a Single Market

Again, we can now just remove the logic that is in the controller.

Resources/views/markets/show.blade.php

```
@extends('layouts.app')
@section('main')

@php
  $market = Market::find(3);
@endphp

  <h1>{{ $market->name }}</h1>
  <h3>{{ $market->city }}</h3>
  site:
  <a href="{{ $market->site }}">
    {{ $market->site }}
  </a>
```

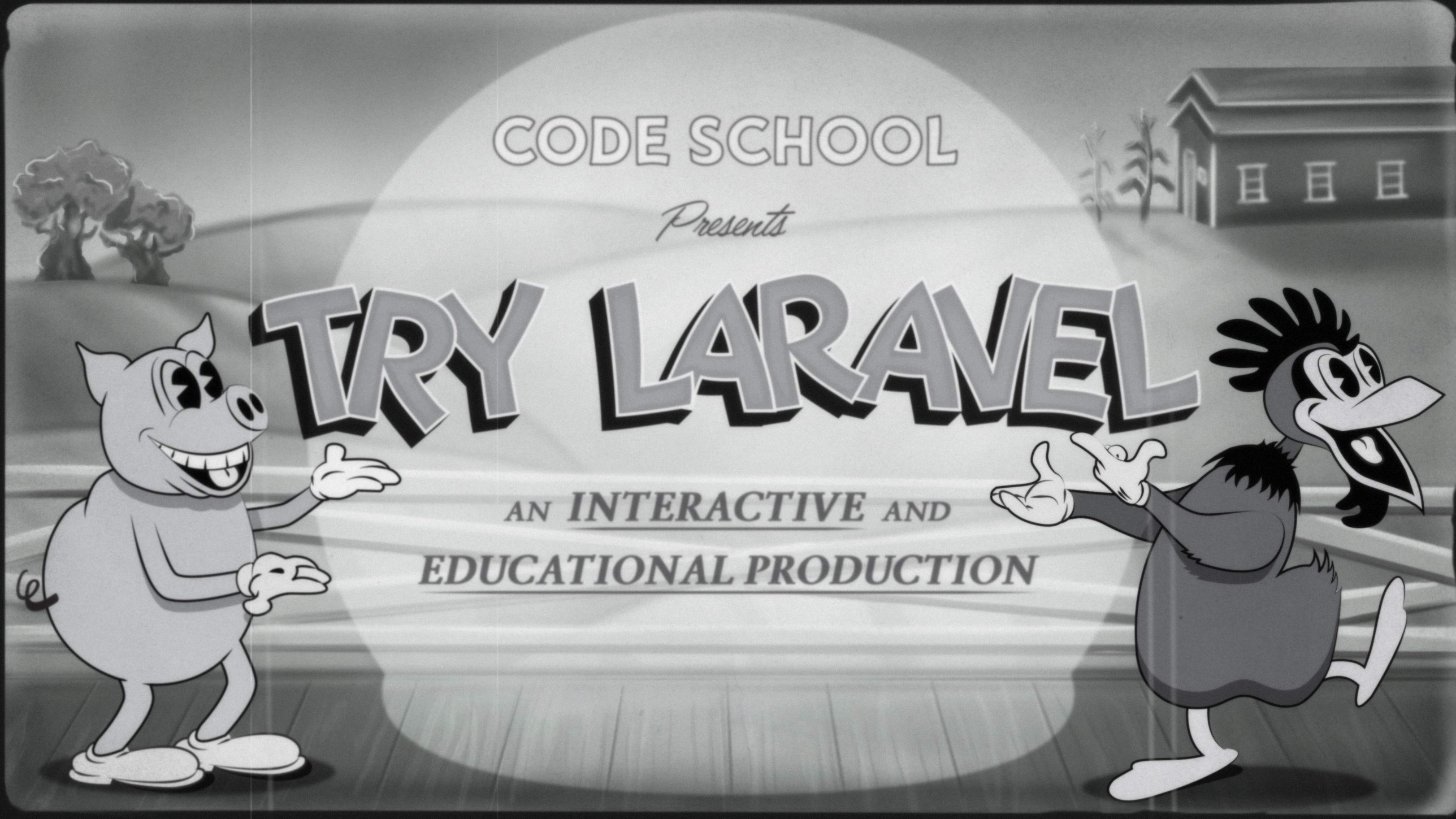
Markets are now passed from the controller!

Controllers in Review

Using controllers for logic

Using methods to render views

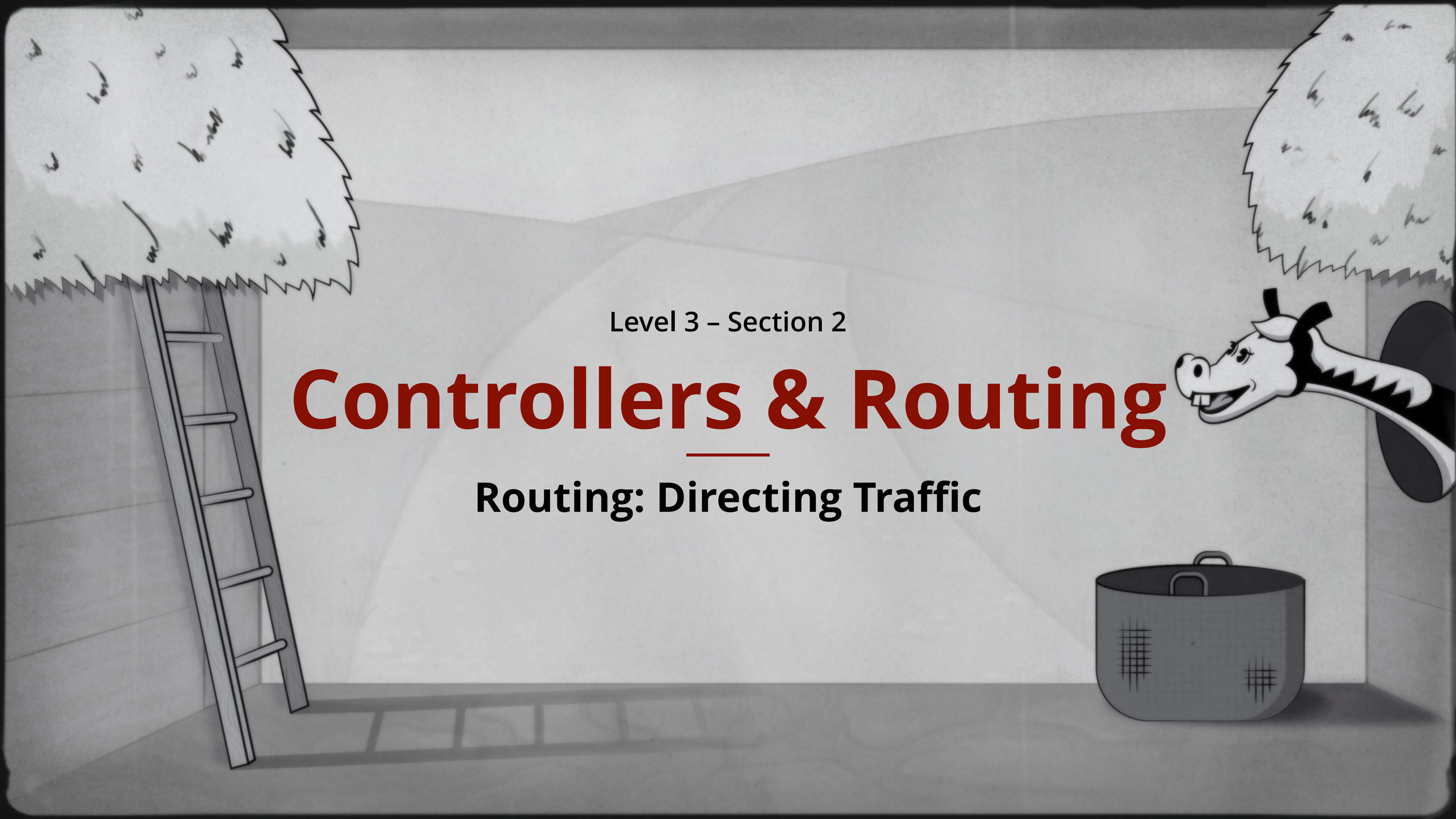
Removing logic from views


CODE SCHOOL
Presents
TRY LARAVEL
AN INTERACTIVE AND
EDUCATIONAL PRODUCTION

Level 3 – Section 2

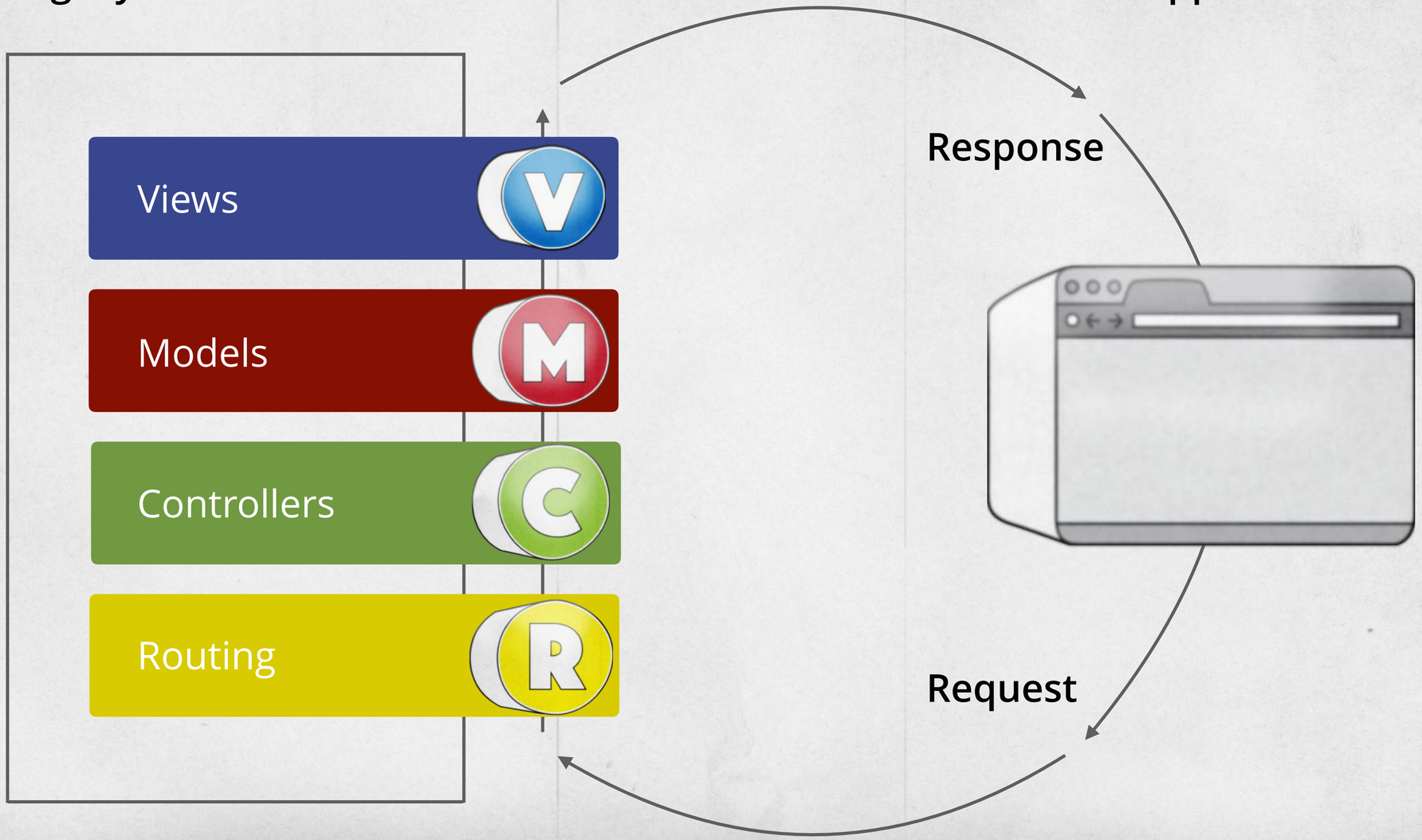
Controllers & Routing

Routing: Directing Traffic



Application Stack Diagram

The routing layer directs our traffic from the user-entered URL to our application.



Routing, Starting Simple

For the most basic of routes, we will accept a location and then return a string.

routes/web.php

```php
<?php

Route::get('/', function () {
  return 'Hello World';
});

Route::get('markets/{id}', function ($id) {
  return 'Requested Market id = ' . $id;
});
```

The first argument is the URL we are trying to match

The second argument is a callback, or the code that will run if the URL is matched

Routing, With Controllers

Since we are using controllers, we will map routes to controller actions.

routes/web.php

```php
<?php

Route::get('/', 'MarketController@index');
Route::get('markets', 'MarketController@index');
```

app/Http/Controllers/MarketController.php

```php
Class MarketController extends Controller
{
    // Show All Markets
    public function index()
    {

    }
}
```

Routing, With Controllers

Since we are using controllers, we will map routes to controller actions.

routes/web.php

```
<?php

Route::get('markets/create', 'MarketController@create');
Route::post('markets', 'MarketController@store');

Route::get('markets/{market}', 'MarketController@show');
```

market.dev/markets/create

market.dev/markets/3

Routing, With Controllers

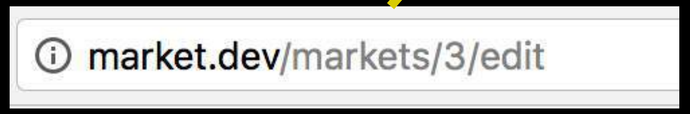
Since we are using controllers, we will map routes to controller actions.

routes/web.php

```
<?php

Route::get('markets/{market}/edit', 'MarketController@edit');
Route::patch('markets/{market}', 'MarketController@update');

Route::delete('markets/{market}', 'MarketController@destroy');
```



Routing: One Line, Many Routes

With the one line, we now have many different routes!

routes/web.php

```php
<?php

Route::get('/', 'MarketController@index');
Route::resource('markets', 'MarketController');
```

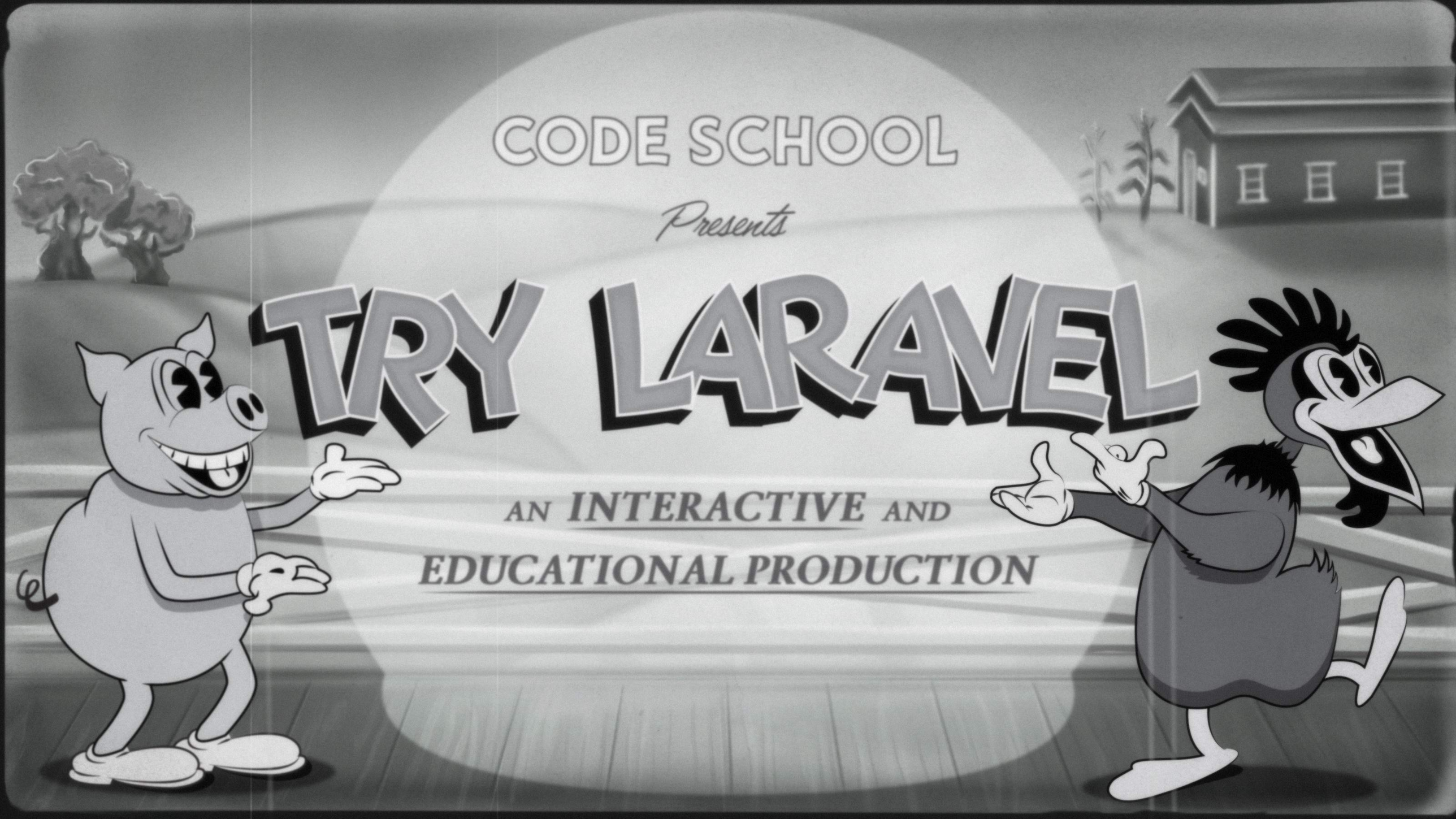
Method	URI	Name	Action
GET	markets	markets.index	MarketController@index
GET	markets/create	markets.create	MarketController@create
POST	markets	markets.store	MarketController@store
DELETE	markets/{market}	markets.destroy	MarketController@destroy
PUT / PATCH	markets/{market}	markets.update	MarketController@update
GET	markets/{market}	markets.show	MarketController@show
GET	markets/{market}/edit	markets.edit	MarketController@edit

Routing in Review

Routing with closures

CRUD routes

Resource routing for models and controllers


CODE SCHOOL
Presents
TRY LARAVEL
AN INTERACTIVE AND
EDUCATIONAL PRODUCTION