

NOW IN

TECHNICOLOR!

From

FORM - TO - TABLE

With

LARAVEL



Level 4

Forms & Validation

Layouts, Forms, & CSRF

Layouts, Forms, & CSRF

What are the steps to continue our CRUD actions?

Create a layout to DRY up our code

Create a show method and template

Create our create and new methods

Code a form in blade to create new markets

D.R.Y. up Our Code with Layouts

Creating a layout for our application will help to 'Don't Repeat Yourself' with duplicate code.

resources/views/layouts/app.blade.php

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

```
    @yield('main')
```

```
</body>
</html>
```

*Create a new folder called layouts and a file named
app.blade.php*

*Using a @yield tag we will define our 'main'
content area for the application*

Modify Our Index Page for the Layout

Removing everything except for the list content, we can clean up our index.blade.php file.

resources/views/markets/index.blade.php

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

Extend our layout file to use the HTML in app.blade.php in the layouts folder

```
<ul>
```

```
    @foreach( $markets as $market )
```

```
        ...
```

```
    @endforeach
```

```
</ul>
```

All content for the index page should fall within the @section & @endsection tags

```
@endsection
```


Create a Show Template Using our Layout

Add an our layout tag and the section tags, then echo out our fields for the \$market.

resources/views/markets/show.blade.php

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

<h1>{{ $market->name }}</h1>

<ul>
    <li>Location City: {{ $market->city }}</li>
    <li>Website: {{ $market->website }}</li>
</ul>

@endsection
```

The \$market object will come into our view from the controller show action

Add a Show Method to Our Market Controller

Adding in a show method will allow us to query by the market object and display our data.

app/Http/Controllers/MarketController.php

```
<?php
class MarketController extends Controller
{
    ...
    public function show(Market $market)
    {
        return view('markets.show', [ 'market' => $market ] );
    }
    ...
}
```

*Laravel will automatically find the
\$market object in the database
or return a not found error*

pass the market object into our view as \$market

Add a Create Method to Our Market Controller

The create method is very simple. It will help us display a form to create a new market.

app/Http/Controllers/MarketController.php

```
<?php
class MarketController extends Controller
{
    ...
    public function create()
    {
        return view('markets.create');
    }
    ...
}
```

*Only a view is required for the
create action*



Make a Create Template Using Our Layout

Add a our layout tag and the section tags, then echo out our fields for the \$market.

resources/views/markets/show.blade.php

```
...
<form action="{{ route('markets.store') }}" method="post">
    <label for="name">Market Name</label>
    <input type="text" name="name">
    <label for="city">Market Name</label>
    <input type="text" name="city">
    <label for="website">Market Name</label>
    <input type="text" name="website">
    <button type="submit">Create</button>
</form>
@endsection
```

The route helper will allow us to use a named route, generated by our resource route, to submit our form to

Modify the Store Method in the Controller

The store method will process our form and add the new data to the database.

app/Http/Controllers/MarketController.php

```
<?php
class MarketController extends Controller
{
    ...
    public function store(Request $request)
    {
        Market::create($request->all());
        return redirect('markets');
    }
    ...
}
```

A request object will contain all of the \$_POST data from our form which we can pass to the create method on Market

Then, redirect to the markets index action

Protect against Cross-site Request Forgery

Protecting against CSRF in Laravel is simple. We only need to add one line of code.

resources/views/markets/show.blade.php

```
<form action="{{ route('markets.store') }}" method="post">
    {{ csrf_field() }}
```

```
<input type="hidden" name="_token" value="DaBSErIR..WPPI">
```

```
<label for="name">Market Name</label>
```

```
<input type="text" name="name">
```

```
<label for="city">Market Name</label>
```

```
<input type="text" name="city">
```

```
<label for="website">Market Name</label>
```

```
<input type="text" name="website">
```

```
<button type="submit">Create</button>
```

```
</form>
```

```
@endsection
```

*csrf_field will generate,
and be replaced by the
following code*

Layouts, Forms, & CSRF Overview

What are the steps to continue our CRUD actions?

Created a layout to DRY up our code

Created a show method and template

Created our create and new methods

Coded a form in blade to create new markets

Protected against cross-site request forgery with a helper



Level 4

Forms & Validation

Pagination & Validation

Results Pagination & Validation

What steps do we need to take to separate all our markets into pages and validate our inputs?

Modify our index action query and sort results

Add pagination to our index action

Add pagination links to our index template

Add validation key value pairs for each input

Add error display if validation fails

Pagination in the Market Controller

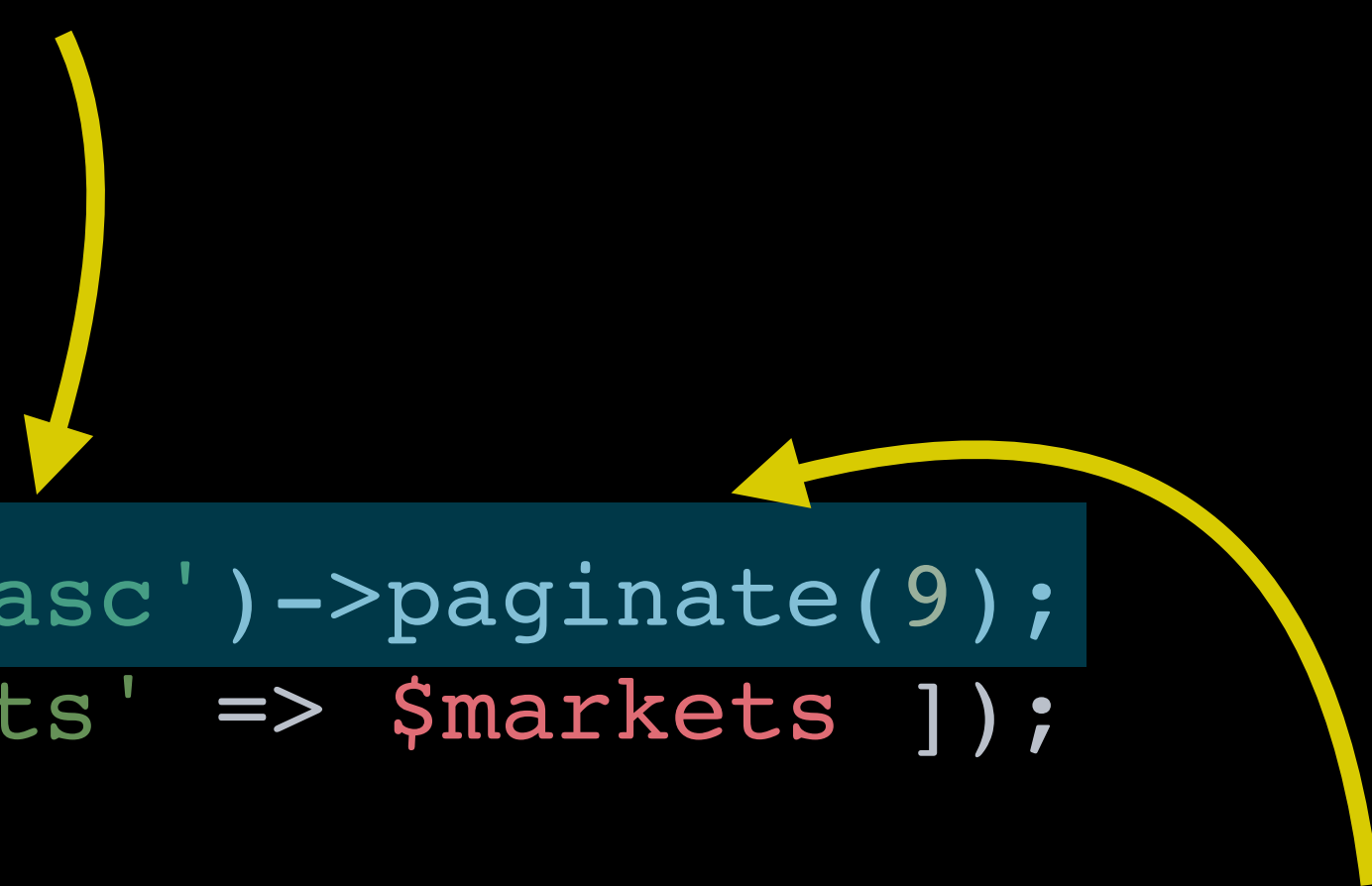
Using the index method, we can setup pagination of our market results.

app/Http/Controllers/MarketController.php

```
<?php
...
class MarketController extends Controller
{
    public function index()
    {
        $markets = Market::orderBy('name', 'asc')->paginate(9);
        return view('markets.index', [ 'markets' => $markets ] );
    }
}
...
```

the orderBy query will return all results sorted by name ascending

The paginate(9) method will break our markets into groups of 9



Using the Route Helper for Each Market

The route helper will assist us in creating a url for the show page of each market.

resources/views/markets/index.blade.php

```
<ul>
  @foreach($markets as $market)
    <li>
      <a href="{{ route('markets.show', $market) }}">
        {{ $market->name }}
      </a>
    </li>
  @endforeach
</ul>
{{ $markets->links() }}
```

*With a second argument to the route helper,
we can pass a market object and its data*



once we add styles, links() will render something like this!



Adding Validation Before Storing Our Market

The store method will process our form and add the new data to the database.

app/Http/Controllers/MarketController.php

```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required|unique:markets|max:255',
        'website' => 'required',
        'city' => 'required',
    ]);

    Market::create($request->all());
    return redirect('markets');
}
```


Display Any Errors on Our Layout

So that we might reuse our code for other models, we can place our errors on the layout.

resources/views/layouts/app.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head><title>Farm To Market</title></head>
<body>
    @if(count($errors) > 0)
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    @endif
```

...

Exit Check on First Error

If we want to exit our validation routine on any errors, add the bail option to your values.

app/Http/Controllers/MarketController.php

```
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'bail|required|unique:markets|max:255',
        'website' => 'bail|required',
        'city' => 'bail|required',
    ]);
    Market::create($request->all());
    return redirect('markets');
}
```

bail will cause the whole validation to stop running if any field is not valid

Pagination & Validation Overview

Review how to paginate our results and add validation before creation.

Modified our index action query and sorted results

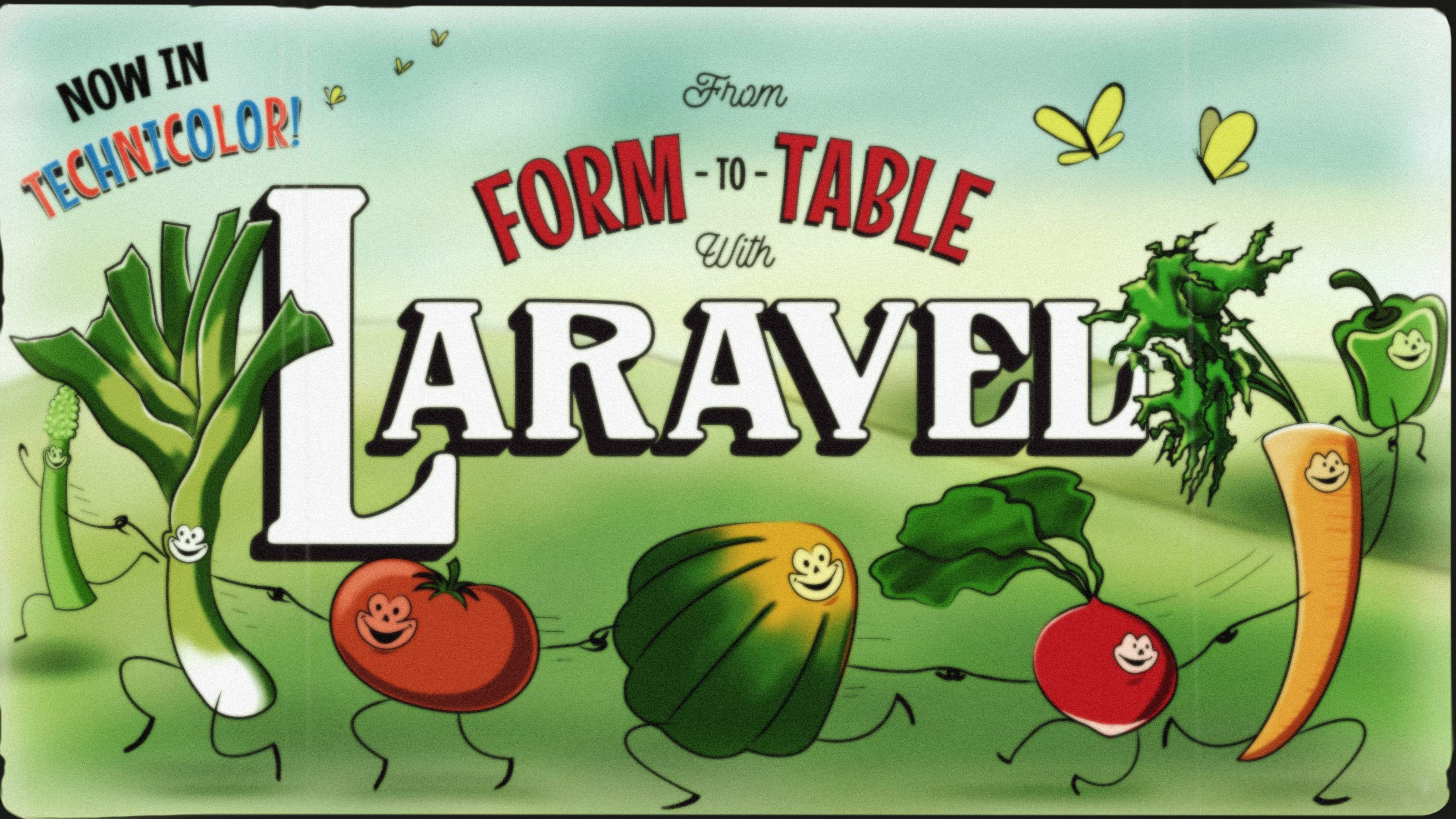
Added pagination to our index action

Added pagination links to our index template

Added validation key value pairs for each input

Added error display if validation fails

Used bail to exit on the first error for each input



NOW IN
TECHNICOLOR!

From

FORM - TO - TABLE
With

LARAVEL