CrestApps /
laravel-code-generator

Code Issues 38 Pull requests 1 Actions Projects Security

An intelligent code generator for Laravel framework that will save you time! This awesome tool will help you generate resources like views, controllers, routes, migrations, languages and/or form-requests! It is extremely flexible and customizable to cover many of the use cases. It is shipped with cross-browsers compatible template, along with a ...

crestapps.com/laravel-code-generator/docs/2.2

MIT license

672 stars 144 forks 38 watching Activity

Public repository

master

Branches Tags

MikeAlhayek ... 15 hours ago

View code

README.md

Tweet

Introduction

An intelligent code generator for Laravel framework that will save you time! This awesome tool will help you generate resources like views, controllers, routes, migrations, languages and/or form-requests! It is extremely flexible and customizable to cover many on the use cases. It is shipped with cross-browsers compatible template, along with a client-side validation to modernize your application.

For full documentation and live demo please visit CrestApps.com

**Note: ** The available documentation is for versions <= 2.2. The documentation for vertion 2.3+ is very similar with some exceptions. Please review the Change Log to get a list of the changes.

Features

- One step installation when using Laravel 5.5+
- Create very clean, reusable and highly readable code to build on.
- Create full resources using a single command with **migration** or from **existing database**.
- Creates full resources for all of the existing tables in the database using one command.
- Create full API-based resources using a single command with **migration** or from **existing database**.
- Create beautiful documentation for your API.
- Create api-resource and api-resource-collection with Laravel 5.5+.
- Allows you to save the fields in a JSON file and recreate resources when the business needs changes.
- Utilizes JSON based resource-file to allow you to define your resources. Resource-file allows you to easily regenerate the resource at any time even when the business rules change.
- Create standard CRUD controllers with simple or form-request validation.
- Customizable view’s templates to enable you to change the standard look and feel of your application.
- Create model with relations.
- Create named routes with and without group.
- Create standard CRUD views.
- Smart migration engine! Keeps track of all generated migrations to only create the needed migration.
- Intelligent enough to automatically handles the relations between the models.
- Very flexible and rich with configurable options.
- Easy commands to create resource-file; additionally, add or reduce existing resource-file.
- Full capability to generate multi-languages applications.
- Client-side validation.
- File uploading handling.
- Auto store multiple-response in the database.
- Create form-request to clean up your controller and increase your code reusability.
- Create view's layouts with and without client-side validation.
- Change the template at run time to generate different views.
- Ability to generate views with and without Laravel-Collective.
- Nicely handles any date, time or datetime field.
- Auto handles any boolean field.

- Very easy to use with lots of documentation.

Installation

1. To download this package into your laravel project, use the command-line to execute the following command

```
composer require crestapps/laravel-code-generator --dev
```

2. **(You may skip this step when using Laravel >= 5.5)** To bootstrap the packages into your project while using command-line only, open the app/Providers/AppServiceProvider.php file in your project. Then, add the following code to the register() method.

   Add the following line to bootstrap laravel-code-generator to the framework.

```
if ($this->app->runningInConsole()) {
    $this->app-
>register('CrestApps\CodeGenerator\CodeGeneratorServiceProvider');
}
```

> A layout is required for the default views! The code generator allows you to create a layout using the command-line. Of cource you can use your own layout. You'll only need to include CSS bootstrap framework in your layout for the default templates to work properly. Additionally, you can chose to you design your own templetes using a different or no css framework.

Lessons

Checkout our channel on YouTube.com

- https://youtu.be/l21qNcsMAWg
- https://youtu.be/infoecfXOCw

Available Commands

> The command in between the square brackets **[]** must be replaced with a variable of your choice.

- **Main commands**
  - php artisan create:scaffold [model-name]

  - php artisan create:controller [model-name]
  - php artisan create:model [model-name]
  - php artisan create:form-request [model-name]
  - php artisan create:routes [model-name]
  - php artisan create:migration [model-name]
  - php artisan create:language [model-name]
  - php artisan create:mapped-resources
- API commands
  - php artisan create:api-scaffold [model-name]
  - php artisan create:api-controller [model-name]
  - php artisan create:api-resources [model-name]
  - php artisan api-doc:create-controller [model-name]
  - php artisan api-doc:create-view [model-name]
- Views commands
  - php artisan create:layout [application-name]
  - php artisan create:views [model-name]
  - php artisan create:index-view [model-name]
  - php artisan create:create-view [model-name]
  - php artisan create:edit-view [model-name]
  - php artisan create:show-view [model-name]
  - php artisan create:form-view [model-name]
- Resource's files commands
  - php artisan resource-file:from-database [model-name]
  - php artisan resource-file:create [model-name]
  - php artisan resource-file:append [model-name]
  - php artisan resource-file:reduce [model-name]
  - php artisan resource-file:delete [model-name]
- Migration commands
  - php artisan migrate-all
  - php artisan migrate:rollback-all
  - php artisan migrate:reset-all
  - php artisan migrate:refresh-all
  - php artisan migrate:status-all

> Full documentation available at CrestApps.com.

> Live demo is available at CrestApps.com.

Upgrading from version <= 2.2 to 2.3+

- Delete the `codegenerator.php` file found in your `config` folder, then rename the `codegenerator_custom.php` file to `laravel-code-generator.php` if one exists. Alternatively, you can delete both `codegenerator.php` and `codegenerator_custom.php`

Contribution

Do you like this project and want to contribute?

- **HELP WANTED** Version `v2.3` needs to be documented. If you are able to contribute, please read the change-log in v2.3 branch and document it in the CrestApps-site repository. For any help, my email can be found in the `composer.json` file, feel free to send me an email.
- **HELP WANTED** Need to create a new theme for Bootstrap 5 and make it the default. If interested, please submit a PR.
- Please start by ***Staring*** this package on GitHub.
- Sharing this projects with others is your way of saying keep improvements and new awesome feature coming.
- Report any bugs or send us any comments, idea, thought that you may have about this project as an issue on GitHub.

What did you create with this package?

I'd love to know if your site was generated using this package and list your logo on the documentation site. Please email using my contact info found in `composer.json` file.

Examples

The following example assumes that we are trying to create a CRUD called *AssetCategory* with the fields listed below.

- id
- name
- description
- is_active

Basic example - CRUD with migration

```
php artisan resource-file:create AssetCategory --
fields=id,name,description,is_active
```

> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*
>
> ```
> php artisan create:scaffold AssetCategory --with-migration
> ```
>
> The above command will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views, the routes, and migration class!*

Basic example - CRUD with migration - Shortcut

> ```
> php artisan create:scaffold AssetCategory --with-migration --fields=id,name,description,is_active
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json* first. Then, it will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views, the routes, and migration class!*. This is a short way to issuing both `resource-file:create` and `create:scaffold` in one line

Basic API example - CRUD with migration

> ```
> php artisan resource-file:create AssetCategory --fields=id,name,description,is_active
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*
>
> ```
> php artisan create:scaffold AssetCategory --with-migration
> ```
>
> The above command will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views, the routes, and migration class!*

Basic example using translations for English and Arabic - with migration

> ```
> php artisan resource-file:create AssetCategory --fields=id,name,description,is_active --translation-for=en,ar
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*
>
> ```
> php artisan create:scaffold AssetCategory --with-migration
> ```
>
> The above command will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views, the routes, and migration class!*

Basic example with form-request

> ```
> php artisan resource-file:create AssetCategory --fields=id,name,description,is_active
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*
>
> ```
> php artisan create:scaffold AssetCategory --with-form-request
> ```
>
> The above command will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views, the routes, and app/Http/Requests/AssetCategoriesFormRequest class!*

Basic example with soft-delete and migration

> ```
> php artisan resource-file:create AssetCategory --fields=id,name,description,is_active
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*
>
> ```
> php artisan create:scaffold AssetCategory --with-soft-delete --with-migration
> ```
>
> The above command will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views, the routes, and migration file!*

Creating resources from existing database

> ```
> php artisan create:scaffold AssetCategory --table-exists
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*. It is going to assume that the table name is called "asset_categories" in your database. If that is not the case, you can use *--table-name=some_other_table_name*
>
> Then it will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views and the routes!*
>
> You may also create a resource-file from existing database separately using `php artisan resource-file:from-database AssetCategory`

Creating resources from existing database with translation for English and Arabic

> ```
> php artisan create:scaffold AssetCategory --translation-for=en,ar --table-exists
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*

> Then it will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views and the routes!*
>
> You may also create a resource-file from existing database separately using `php artisan resource-file:from-database AssetCategory --translation-for=en,ar`

Creating resources from existing database with translation for English and Arabic in two step for better control over the fields!

> ```
> php artisan resource-file:from-database AssetCategory --translation-for=en,ar
> ```
>
> ```
> php artisan create:scaffold AssetCategory
> ```
>
> The above command will create resource-file names */resources/laravel-code-generator/sources/asset_categories.json*
>
> Then it will create a model *app/Models/AssetCategory*, a controller *app/Http/Controllers/AssetCategoriesController, all views and the routes!*

What's new?

- Release Notes
- Upgrade Guide

License

"Laravel Code Generator" is an open-sourced software licensed under the MIT license

Releases 44

v2.4.9 Latest
on Sep 9, 2022

+ 43 releases

Sponsor this project

Packages

No packages published

Used by 270

+ 262

Contributors 13

+ 2 contributors

Languages

● **PHP** 100.0%