


<div><div><div><div><div><div></div><div>tihomiro</div></div></div><div>Merge pull request #206 from akoevroman/bugfix/204</div><div>11f4355 · on Feb 11</div><div>🕒 152 commits</div></div></div></div>		
src	Add Database\Seeders namespace to seed stub	2 years ago
tests	Add Database\Seeders namespace to seed stub	2 years ago
.gitignore	Update .gitignore	3 years ago
.travis.yml	Update .travis.yml	2 years ago
LICENSE	Implemented a composer dump-autoload feature	6 years ago
README.md	Merge branch 'master' of github.com:orangehill/iseed	3 years ago
composer.json	this commit resolve #204	7 months ago
phpunit.xml	upgraded	3 years ago

📄 README.md

Inverse seed generator (iSeed) is a Laravel package that provides a method to generate a new seed file based on data from the existing database table.

🔗 build passing

🔗 stable v3.0.2

📄 downloads 3.78 M

📊 Analytics

Installation

1. Require with Composer

```
composer require orangehill/iseed
```

Laravel 5.3.7 and below or Laravel 4 need specific version

```
composer require orangehill/iseed:2.2 # Laravel 5.3.7 and below
composer require orangehill/iseed:1.1 # Laravel 4
```

2. Add Service Provider (Laravel 5.4 and below)

Latest Laravel versions have auto discovery and automatically add service provider - if you're using 5.4.x and below, remember to add it to providers array at /app/config/app.php:

```
// ...
Orangehill\Iseed\IseedServiceProvider::class,
```

Artisan command options

[table_name]

Mandatory parameter which defines which table/s will be used for seed creation. Use CSV notation for multiple tables. Seed file will be generated for each table.

Examples:

```
php artisan iseed my_table
```

```
php artisan iseed my_table,another_table
```

classnameprefix & classnamesuffix

Optionally specify a prefix or suffix for the Seeder class name and file name. This is useful if you want to create an additional seed for a table that has an existing seed without overwriting the existing one.

Examples:

```
php artisan iseed my_table --classnameprefix=Customized
```

outputs CustomizedMyTableSeeder.php

```
php artisan iseed my_table,another_table --classnameprefix=Customized
```

outputs CustomizedMyTableSeeder.php and CustomizedAnotherTableSeeder.php

```
php artisan iseed my_table --classnamesuffix=Customizations
```

outputs MyTableCustomizationsSeeder.php

```
php artisan iseed my_table,another_table --classnamesuffix=Customizations
```

outputs MyTableCustomizationsSeeder.php and AnotherTableCustomizationsSeeder.php

force

Optional parameter which is used to automatically overwrite any existing seeds for desired tables

Example: The following command will overwrite UsersTableSeeder.php if it already exists in laravel's seeds directory.

```
php artisan iseed users --force
```

dumpauto

Optional boolean parameter that controls the execution of composer dump-autoload command. Defaults to true.

Example that will stop composer dump-autoload from execution:

```
php artisan iseed users --dumpauto=false
```

clean

Optional parameter which will clean app/database/seeds/DatabaseSeeder.php before creating new seed class.

Example:

```
php artisan iseed users --clean
```

database

Optional parameter which specifies the DB connection name.

Example:

```
php artisan iseed users --database=mysql2
```

max

Optional parameter which defines the maximum number of entries seeded from a specified table. In case of multiple tables, limit will be applied to all of them.

Example:

```
php artisan iseed users --max=10
```

chunksize

Optional parameter which defines the size of data chunks for each insert query.

Example:

```
php artisan iseed users --chunksize=100
```

orderby

Optional parameter which defines the column which will be used to order the results by, when used in conjunction with the max parameter that allows you to set the desired number of exported database entries.

Example:

```
artisan iseed users --max=10 --orderby=id
```

direction

Optional parameter which allows you to set the direction of the ordering of results; used in conjunction with orderby parameter.

Example:

```
artisan iseed users --max=10 --orderby=id --direction=desc
```

exclude

Optional parameter which accepts comma separated list of columns that you'd like to exclude from tables that are being exported. In case of multiple tables, exclusion will be applied to all of them.

Example:

```
php artisan iseed users --exclude=id
php artisan iseed users --exclude=id,created_at,updated_at
```

prerun

Optional parameter which assigns a laravel event name to be fired before seeding takes place. If an event listener returns false, seed will fail automatically. You can assign multiple preruns for multiple table names by passing an array of comma separated DB names and respectively passing a comma separated array of prerun event names.

Example: The following command will make a seed file which will fire an event named 'someEvent' before seeding takes place.

```
php artisan iseed users --prerun=someEvent
```

The following example will assign someUserEvent to users table seed, and someGroupEvent to groups table seed, to be executed before seeding.

```
php artisan iseed users,groups --prerun=someUserEvent,someGroupEvent
```

The following example will only assign a someGroupEvent to groups table seed, to be executed before seeding. Value for the users table prerun was omitted here, so users table seed will have no prerun event assigned.

```
php artisan iseed users,groups --prerun=someGroupEvent
```

postrun

Optional parameter which assigns a laravel event name to be fired after seeding takes place. If an event listener returns false, seed will be executed, but an exception will be thrown that the postrun failed. You can assign multiple postruns for multiple table names by passing an array of comma separated DB names and respectively passing a comma separated array of postrun event names.

Example: The following command will make a seed file which will fire an event named 'someEvent' after seeding was completed.

```
php artisan iseed users --postrun=someEvent
```

The following example will assign someUserEvent to users table seed, and someGroupEvent to groups table seed, to be executed after seeding.

```
php artisan iseed users,groups --postrun=someUserEvent,someGroupEvent
```

The following example will only assign a someGroupEvent to groups table seed, to be executed after seeding. Value for the users table postrun was omitted here, so users table seed will have no postrun event assigned.

```
php artisan iseed users,groups --postrun=someGroupEvent
```

noindex

By using --noindex the seed can be generated as a non-indexed array. The use case for this feature is when you need to merge two seed files.

Example:

```
php artisan iseed users --noindex
```

Usage

To generate a seed file for your users table simply call: \Iseed::generateSeed('users', 'connectionName', 'numOfRows');. connectionName and numOfRows are not required arguments.

This will create a file inside a /database/seeds (/app/database/seeds for Laravel 4), with the contents similar to following example:

```
<?php

// File: /database/seeds/UsersTableSeeder.php

class UsersTableSeeder extends Seeder {

    /**
     * Auto generated seed file
     *
     * @return void
     */
    public function run()
    {
        \DB::table('users')->truncate();
        \DB::table('users')->insert(array (
            0 =>
                array (
                    'id' => '1',
                    'email' => 'admin@admin.com',
                    'password' => '$2y$10$tUGCKqf/8NY3w119sobGsudt6UngnoVxX/1Uoh9E1c5008ERRkK9C',
                    'permissions' => NULL,
                    'activated' => '1',
                    'activation_code' => NULL,
                    'activated_at' => NULL,
                    'last_login' => NULL,
                    'persist_code' => NULL,
                    'reset_password_code' => NULL,
                    'first_name' => NULL,
                    'last_name' => NULL,
                    'created_at' => '2013-06-11 07:47:48',
                    'updated_at' => '2013-06-11 07:47:48',
                ),
            1 =>
                array (
                    'id' => '2',
                    'email' => 'user@user.com',
                    'password' => '$2y$10$1mVvsHzK/80gnSYgpjs/30jMKMHeA9BH/hj143E1uBuLk2GPMuZ2W',
                    'permissions' => NULL,
                    'activated' => '1',
                    'activation_code' => NULL,
                    'activated_at' => NULL,
                    'last_login' => '2013-06-11 07:54:57',
                    'persist_code' => '$2y$10$C01a8UuyqC6AU2TPlwJ0I.E3Mr=va8A3tuVFwXxNSu7j5wRkzsYYHK',
                    'reset_password_code' => NULL,
                    'first_name' => NULL,
                    'last_name' => NULL,
                    'created_at' => '2013-06-11 07:47:48',
                    'updated_at' => '2013-06-11 07:54:57',
                ),
            ));
    }
}
```

This command will also update /database/seeds/DatabaseSeeder.php (/app/database/seeds/DatabaseSeeder.php for Laravel 4) to include a call to this newly generated seed class.

If you wish you can define custom iSeed template in which all the calls will be placed. You can do this by using #iseed_start and #iseed_end templates anywhere within /database/seeds/DatabaseSeeder.php (/app/database/seeds/DatabaseSeeder.php for Laravel 4), for example:

```
<?php

// File: /database/seeds/DatabaseSeeder.php
class DatabaseSeeder extends Seeder {

    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        Eloquent::unguard();

        if(App::environment() == "local")
        {
            throw new \Exception('Only run this from production');
        }

        #iseed_start

        // here all the calls for newly generated seeds will be stored.

        #iseed_end
    }
}
```

Alternatively you can run Iseed from the command line using Artisan, e.g. php artisan iseed users. For generation of multiple seed files comma separated list of table names should be send as an argument for command, e.g. php artisan iseed users,posts,groups.

In case you try to generate seed file that already exists command will ask you a question whether you want to overwrite it or not. If you wish to overwrite it by default use --force Artisan Command Option, e.g. php artisan iseed users --force.

If you wish to clear iSeed template you can use Artisan Command Option --clean, e.g. php artisan iseed users --clean. This will clean template from app/database/seeds/DatabaseSeeder.php before creating new seed class.

You can specify db connection that will be used for creation of new seed files by using Artisan Command Option --database=connection_name, e.g. php artisan iseed users --database=mysql2.

To limit number of rows that will be exported from table use Artisan Command Option --max=number_of_rows, e.g. php artisan iseed users --max=10. If you use this option while exporting multiple tables specified limit will be applied to all of them.

To (re)seed the database go to the Terminal and run Laravel's db:seed command (php artisan db:seed).

Please note that some users encountered a problem with large DB table exports (error when seeding from table with many records). The issue was solved by splitting input data into smaller chunks of elements per insert statement. As you may need to change the chunk size value in some extreme cases where DB table has a large number of columns, the chunk size is configurable in iSeed's config.php file:

```
'chunk_size' => 500 // Maximum number of rows per insert statement
```

About

Laravel Inverse Seed Generator

📄 Readme

🔗 BSD-2-Clause license

🌟 2.4k stars

👁 58 watching

🍴 329 forks

Releases 18

🔗 Add Laravel 9 support

Latest

on Feb 13

+ 17 releases

Packages

No packages published

Used by 4.2k



+ 4,215

Contributors 23



+ 12 contributors

Languages

● PHP 100.0%