As a package in an existing Laravel app

Getting a license

If you already have a Laravel 10 application, you can install Mixpost as a package inside it.

To install Mixpost, you'll need to get a license first.

Installation via Composer

First, add the `packages.inovector.com` repository to your `composer.json`.

```
"repositories": [
    {
        "type": "composer",
        "url": "https://packages.inovector.com"
    }
],
```

Next, you need to create a file called `auth.json` and place it either next to the `composer.json` file in your project or the composer home directory. Using this command, you can determine the composer home directory on *nix machines.

```
composer config --list --global | grep home
```

This is the content you should put in `auth.json`:

```
{
    "http-basic": {
        "packages.inovector.com": {
            "username": "<YOUR-MIXPOST.APP-ACCOUNT-EMAIL-ADDRESS-HERE>",
```

```
            "password": "<YOUR-LICENSE-KEY-HERE>"
        }
    }
}
```

To validate if Composer can read your `auth.json` you can run this command:

```
composer config --list --global | grep packages.inovector.com
```

If you did everything correctly, the above command should display your credentials. If that command doesn't display anything, verify that you created an auth.json as mentioned above.

With this configuration in place, you'll be able to install the package into your Laravel project using this command:

```
composer require inovector/mixpost-pro-team "^2.0"
```

After installing the Mixpost package, you may execute:

```
php artisan mixpost:install
```

To ensure that assets get republished each time Mixpost is updated, we strongly advise you to add the following command to the `post-update-cmd` of the `scripts` section of your `composer.json`.

```
"scripts": {
    "post-update-cmd": [
        "@php artisan mixpost:publish-assets --force=true"
    ]
}
```

Mixpost uses Job Batching and you should create a database migration to build a table to contain meta-information about your job batches.

If your application does not yet have this table, it may be generated using the:

```
php artisan queue:batches-table
```

Run the migrations with:

```
php artisan migrate
```

To make media files (images & videos) accessible from the web you should create a symbolic link from public/storage to storage/app/public

```
php artisan storage:link
```

You can publish the config file with:

```
php artisan vendor:publish --tag=mixpost-config
```

Mixpost has the ability to generate images from video while uploading a video file. This would not be possible without FFmpeg installed on your server. You need to follow FFmpeg installation instructions on their official website.

After installation, depending on the operating system, you need to set the `ffmpeg_path` and `ffprobe_path` in the Mixpost config file.

Default folder path: `/usr/bin/`. If FFmpeg is there, there is no need to change it.

```
/*
 * FFMPEG & FFProbe binaries paths, only used if you try to generate video thumbnails
 */
'ffmpeg_path'  => env('FFMPEG_PATH', '/usr/bin/ffmpeg'),
'ffprobe_path'  => env('FFPROBE_PATH', '/usr/bin/ffprobe'),
```

Or, you can set them in your `.env` file

```
FFMPEG_PATH=/usr/bin/ffmpeg
FFPROBE_PATH=/usr/bin/ffprobe
```

Error Reporting

Mixpost utilizes its unique internal exception handler rather than the default "**App\Exceptions\ExceptionHandler**". To integrate external error reporting tools with your Mixpost setup, you should use the "**Mixpost::report**" method. Generally, this method is called from the register method of your app's "**App\Providers\AppServiceProvider**" class:

```
use Inovector\Mixpost\Mixpost;
use Sentry\Laravel\Integration;

Mixpost::report(function($exception) {
   Integration::captureUnhandledException($exception);
});
```

Install Horizon

Mixpost handles various tasks in a queued way via Laravel Horizon. If your application doesn't have Horizon installed yet, follow their installation instructions.

After Horizon is installed, don't forget to set `QUEUE_CONNECTION` in your `.env` file to `redis`.

`config/horizon.php` should have been created in your project. In this config file, you must add a block named `mixpost-heavy` to both the `production` and `local` environment.

```
    'environments' => [
        'production' => [
            'supervisor-1' => [
                'maxProcesses' => 10,
                'balanceMaxShift' => 1,
                'balanceCooldown' => 3,
            ],
            'mixpost-heavy' => [
                'connection' => 'mixpost-redis',
                'queue' => ['publish-post'],
                'balance' => 'auto',
                'processes' => 8,
                'tries' => 1,
                'timeout' => 60 * 60,
            ],
        ],

        'local' => [
            'supervisor-1' => [
                'maxProcesses' => 3,
            ],
            'mixpost-heavy' => [
                'connection' => 'mixpost-redis',
                'queue' => ['publish-post'],
                'balance' => 'auto',
                'processes' => 3,
                'tries' => 1,
                'timeout' => 60 * 60,
            ],
```

```
        ],
    ],
```

In the `config/queue.php` file you must add the `mixpost-redis` connection:

```
'connections' => [

    // ...

   'mixpost-redis' => [
        'driver' => 'redis',
        'connection' => 'default',
        'queue' => env('REDIS_QUEUE', 'default'),
        'retry_after' => 11 * 60,
        'block_for' => null,
    ],
```

Don't forget to run `php artisan horizon`. In production, you need a way to keep your `horizon` processes running. For this reason, you need to configure a process monitor Supervisor that can detect when your `horizon` processes exit and automatically restart them.

Example of supervisor config:

```
[program:mixpost_horizon]
process_name=%(program_name)s
command=php /path-to-your-project/artisan horizon
autostart=true
autorestart=true
user=your_user_name
stopwaitsecs=3600
```

Schedule the commands

In the console kernel (`app/Console/Kernel.php`), you should schedule these commands:

```
protected function schedule(Schedule $schedule)
{
    // ...
    \Inovector\Mixpost\Schedule::register($schedule);
```

```
    $schedule->command('horizon:snapshot')->everyFiveMinutes();
    $schedule->command('queue:prune-batches')->daily();
}
```

Don't forget to add a cron that runs the scheduler:

```
* * * * cd /path-to-your-project && php artisan schedule:run >> /dev/null 2>&1
```

Visit the UI

After performing all these steps, you should be able to visit the Mixpost UI at /mixpost.

Revision #25
Created 1 May 2023 15:08:37 by Dima Botezatu
Updated 15 April 2024 18:48:29 by Dima Botezatu