Adding a class alias at boot time in Laravel

Tom Rochette <tom.rochette@coreteks.org>

March 28, 2021 — 708d870e

I make extensive use of Laravel Debugbar to track performance of parts of my application. I sprinkle calls to `Debugbar::startMeasure` and `Debugbar::stopMeasure` to track the duration of certain segments of my code. However, when this code goes into production, this dependency isn't present. This cause the code to break since it cannot find `Debugbar` anymore.

To solve this issue, I thought I would create a dummy `Debugbar` class and have it added as an alias, so that any code depending on `Debugbar` would still work, but end up as a "no operation". I found the article Dynamic class aliases in package which introduced the necessary piece of information to accomplish this.

```php
<?php

use Illuminate\Foundation\AliasLoader;
use My\SuperPackage\FooBar;

class ServiceProvider extends \Illuminate\Support\ServiceProvider
{
    public function register()
    {
        $this->app->booting(function() {
            $loader = AliasLoader::getInstance();
            $loader->alias('FooBar', FooBar::class);
        });
    }
}
```

In my desired use case, I simply implemented the following changes:

In `app/Providers/DebugbarServiceProvider.php` (a new file)

```php
<?php

namespace App\Providers;

use Illuminate\Foundation\AliasLoader;
use Illuminate\Support\ServiceProvider;

class DebugbarServiceProvider extends ServiceProvider
{
    public function register()
    {
        if (!class_exists('Debugbar')) {
            $loader = AliasLoader::getInstance();
            $loader->alias('Debugbar', NullDebugbar::class);
        }
    }
```

```
}

class NullDebugbar
{
    public static function __callStatic(string $name, array $arguments)
    {
        // Do nothing
    }
}
```

In `app/config/app.php`

```
    // under the 'providers' key, add
    'providers' => [
        [...]
        // This will take care of loading the service provider defined above
        App\Providers\DebugbarServiceProvider::class,
    ],
```

With those two changes, it is now possible to make use of `Debugbar` in most places and have it work even without the Laravel Debugbar dependency installed.

1 References

- https://laracasts.com/discuss/channels/laravel/dynamic-class-aliases-in-package