مقدمة

الجوهرية، Laravel تطبيقك، ككامل خدمات .Laravel مقدمو الخدمات هن الموقع المركزي لكامل عمليات تمهيد تطبيق ؟ نقصد به عمومًا تسجيل الأشياء، بما في (bootstrapping) "مُمَهَّد عبر مقدمي الخدمات. لكن ما الذي نقصده "بالتمهيد مقدمو الخدمات ذلك تسجيل ارتباطات حاويات الخدمات، ومنصتات الأحداث، والبرمجيات الوسيطة، وحتى المسارات (routes). فسترى Laravel المتضمن في config/app.php هي المكان المركزي لإعداد تطبيقك مثلما ذكرنا سابقًا. إن فتحت الملف والتي تمثّل كل أصناف مقدمي الخدمات التي سَتُحَمَّل من أجل تطبيقك. كثير منهن طبعا مقدمون providers، المصفوفة مؤجلون". أي أنهن لن يتُحَمَّلن عند كل طلب بل عند الحاجة إلى خدماتها فقط.

في هذه النظرة العامة ستتعلم كيف تكتب مقدمي خدماتك وتسجلهم في تطبيق Laravel الخاص بك.

كتابة مقدمي الخدمات

يحتوي أغلب مقدمو الخدمات دالة .Illuminate\Support\ServiceProvider تُوسِّع كل مقدمو الخدمات الصنف لا يجب أن تحاول .register يجب ربط الأشياء داخل حاوي الخدمات فقط داخل كل دالة .boot و register تسجيل أي منصت أحداث، مسار أو أي جزء وظيفي آخر ضمن دالة register.

مقدّمًا جديدًا باستخدام الأمر Artisan تولّد واجهة سطر الأوامر لبرمجية make:provider:

```
php artisan make:provider RiakServiceProvider
```

التابع register

لا يجب أن تحاول أبدًا تسجيل .register كما ذُكِر سابقًا، يجب أن تربط الأشياء فقط في حاوي الخدمات في تابع التسجيل أي منصت أحداث، أو مسار، أو أي جزء وظيفي آخر ضمن تابع التسجيل. إن لم تفعل ذلك، فقد تستخدم خدمة لم يُحَمَّل مقدم خدماتها بعد عن غير قصد.

في أي من توابع مقدم الخدمة والتي تمنحك $app فلنلقِ نظرة على مقدم خدمات بسيط. لديك دائما حق الوصول لخاصية القدرة على الوصول لحاوي الخدمات:

```
namespace App\Providers;

use Riak\Connection;
use Illuminate\Support\ServiceProvider;

class RiakServiceProvider extends ServiceProvider
{
    /**
     * تسجيل الارتباطات في الحاوي
     *
     * @return void
     */
    public function register()
    {
        $this->app->singleton(Connection::class, function ($app) {
            return new Connection(config('riak'));
        });
    }
}
```

في حاوي الخدمات. Riak\Connection تُعَرِّف هذه الخدمة دالة التسجيل فقط وتستخدم تلك الدالة لتُعَرِّف تعريف الاستخدام إن لم تفهم كيفية عمل حاوي الخدمات ألْقِ نظرةً على توثيقه.

الخاصيات singletons و bindings

بدل تسجيل singletons و bindings إن سجل مقدم خدمتك عدة ارتباطات بسيطة، فقد ترغب في استخدام الخاصيات كل ارتباط بالحاوي على حدة. عندما يُحَمَّل مقدم الخدمات من طرف إطار العمل سيتفقد هذه الخاصيات تلقائيًا ويسجل ارتباطاتها:

```
namespace App\Providers;

use App\Contracts\ServerProvider;
use App\Contracts\DowntimeNotifier;
use Illuminate\Support\ServiceProvider;
use App\Services\PingdomDowntimeNotifier;
use App\Services\DigitalOceanServerProvider;

class AppServiceProvider extends ServiceProvider
{
    /**
  *. يجب تسجيل كل ارتباطات الحاوي
     *
     * @var array
     */
    public $bindings = [
        ServerProvider::class => DigitalOceanServerProvider::class,
    ];

    /**
     *  الحاوي singletons يجب تسجيل جميع
     * @var array
     */
    public $singletons = [
        DowntimeNotifier::class => PingdomDowntimeNotifier::class,
    ];
}
```

التابع boot

يُنادى boot. داخل مقدم خدماتنا؟ يجب القيام بذلك داخل التابع (view composer) ماذا لو احتجنا لتسجيل مؤلف عرض هذا التابع بعد تسجيل كل مقدمي الخدمات الأخرى، أي عندما يكون لديك حق الولوج لكل خدمة أخرى مُسجَّلة في إطار العمل:

```
namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class ComposerServiceProvider extends ServiceProvider
{
    /**
     * مَهِّد أية خدمات تطبيقية
     *
     * @return void
     */
    public function boot()
    {
        view()->composer('view', function () {
           //
        });
    }
}
```

إضافة الاعتماديَّة للتابع Boot

بمقدم خدماتك. سيضيف حاوي الخدمات أي اعتماديات تحتاج boot الاعتمادية للتابع (type-hint) تستطيع التلميح عن نوع لها تلقائيًا:

```
use Illuminate\Contracts\Routing\ResponseFactory;

public function boot(ResponseFactory $response)
```

```
{
    $response->macro('caps', function ($value) {
        //
    });
}
```

تسجيل المقدِّمين

يتضمن هذا الملف مصفوفة من المُقَدِّمين config/app.php. كل مقدمي الخدمات مُسَجَّلات في ملف الإعدادات providers حيث تستطيع إيراد أسماء أصناف مُقَدِّمي خدماتك. ستجد هناك قائمةً تحتوي أسماء مقدمي الخدمات الأساسية لإطار Laravel. يُمَهِّد هؤلاء المُقَدِّمون مكوِّنات Laravel الجوهرية مثل المُرسِل (mailer) والطابور (bootstrap) افتراضيًا. وغيرها. وذاكرة التخزين المؤقتة (cache)، (queue).

لتُسَجِّل مُقَدِّمك أضفه لهذه المصفوفة:

```
'providers' => [
    // مقدمو الخدمات الأخرى
    App\Providers\ComposerServiceProvider::class,
],
```

المقدِّمون المؤجَّلون

يمكنك تأجيل تسجيل مقدم خدماتك حتى تحتاج لإحدى ارتباطاته المُسجَّلة ان كان يُسجِّل ارتباطاته في حاوي (bindings) الخدمات حصرا. يُحسِّن تأجيل تحميل هذا النوع من المقدمين من أداء تطبيقك بما أنها لا تحمَّل من نظام الملفات مع كل طلب.

مقدم الخدمة Laravel يُترجِم ويُخَزِّن قائمة من كل الخدمات التي توفِّرها مُقدِّمات الخدمات المُؤجَّلات. ثم يُحَمِّل Laravel أحد تلك الخدمات (resolve) فقط عندما تحاول قبول.

يجب على دالة التقديم provides. وعرِّف دالة تقديم true على defer لتأجيل تحميل مقدِّم ما، اضبط خاصية التأجيل provides رد إرتباطات حاوي الخدمات التي سجلها المقدِّم:

```
namespace App\Providers;

use Riak\Connection;
use Illuminate\Support\ServiceProvider;

class RiakServiceProvider extends ServiceProvider
{
    /**
     * تشير إن كان المقدِّم مُؤجَّلا
     *
     * @var bool
     */
    protected $defer = true;

    /**
     * تسجيل لمقدِّم الخدمات
     *
     * @return void
     */
    public function register()
    {
        $this->app->singleton(Connection::class, function ($app) {
            return new Connection($app['config']['riak']);
        });
    }

    /**
```

```
      * الحصول على الخدمة التي يوفرها مقدِّم الخدمات.
      *
      * @return array
      */
     public function provides()
     {
         return [Connection::class];
    }

}
```