Understand service providers in Laravel

Thursday, June 4th, 2015 by Servage

The so called service providers are the heartbeat of your Laravel application. They are the central element of the initialization process where all the relevant and required code is loaded by PHP. This includes all the essentials from the framework itself, but also any own and custom code you need to load.

Bootstrap service providers

During the initializing process the code bootstraps itself. That means it gets ready to go by registering service container bignings, event listeners, middleware, configuration and routes. Service providers are therefore a central place to setup your application. In the config/app.php you will find an array of providers listing all the service provider classes that are loaded during bootstrapping. Beware that many of the service provider classes may be so called deferred providers. It means they are only loaded upon request, and not always included by default. Making a provider deferred works well for classes that only need to be loaded sometimes, because it reduces the performance overhead and load time of your web application.

Create a custom service provider

The code below is an example of a custom service provider. It extends the Illuminate\Support\ServiceProvider class which is an abstract that requires you to define at least a method called “register”. The purpose of the register method is to bind values in the service container.

```
<?php namespace App\Providers;

use Illuminate\Support\ServiceProvider;

class MyServiceProvider extends ServiceProvider {

    /**
     * Register bindings in the container.
```

```
     *
     * @return void
     */
    public function register()
    {
            // Your code ...
    }

}

?>
```

The service container can also have a method called “boot” which executes after all service providers have been made available. Therefore you can use any of the other service providers in the boot method. Note below that the boot method supports type-hinted dependency injection.

```
<?php namespace App\Providers;

use Event;
use Illuminate\Support\ServiceProvider;

class MyServiceProvider extends ServiceProvider {

    /**
     * Perform post-registration booting of services.
     *
     * @return void
     */
    public function boot()
    {
                // Your code ...
    }

    /**
     * Register bindings in the container.
     *
     * @return void
     */
    public function register()
    {
            // Your code ...
    }

}
```

```
?>
```

Register a custom service provider

Now that you have created a custom service provider, you need to register it like the default framework code during the boostrapping process. You can do so in the config/app.php where you need to add your custom service provider to the array like in the example below:

```
'providers' => [
    // Other Service Providers
    'App\Providers\MyServiceProvider',
],
```

Defered service providers

Previously deferred service providers were mentioned to be providers that only load upon request. You can make your service provider deferred by adding it as a boolean flag to the service provider class. This avoids the provider loading all bindings from the filesystem and thus increases performance.

```
/**
 * Indicates if loading of the provider is deferred.
 *
 * @var bool
 */
protected $defer = true;
```

Overall service providers offer a good way to structure and centralize code in Laravel. It offers you a convenient way to provide a specific “service” to any method in your codebase.