In this tutorial, I'll take you through an example on how to create a Laravel package in just 7 steps. So, let's go ahead and dive into it.

1. Create our folder for our new package.

Create a fresh Laravel project;

```
laravel new core
```

After a new Laravel install we create a new folder called packages inside the root of our app.

We are going to use this packages folder instead of the vendor folder because files inside of our vendor folder should only be managed by composer. Once we have published our package we can then include it in our vendor folder using composer.

Next, we will create a folder inside of packages which will be the name of our package **calculator**.

Lastly we will add one last folder called `src` which is where the code for our pacakge will live.

So, our folder structure will be `packages /calculator/src`

Ok, and now that we have our folders ready, let's go ahead and name our package in our project by initializing a `composer.json` file.

2. Name our package and initialize it with composer.

Inside your command prompt navigate to the folder with your package name. In our case: `packages/calculator`, and run the following command:

```
composer init
```

This will initialize the current folder as a composer package and it will ask you a series of questions to setup your package. It will ask for the name, description, author, and a little more info about your package. After you have finished this onscreen setup you will have a new file in your package folder called `composer.json`.

The composer.json file will look similar to the following:

```
{
“name”: “calculator”,
“description”: “This demo package will do some calculations”,
“authors”: [
{
“name”: “YOUR NAME”,
“email”: “YOUR EMAIL”
}
],
“minimum-stability”: “dev”,
“require”: {},

}
```

Great! Now that we have initialized our package we need to tell our main app `composer.json` file to load it.

3. Load our package from our application composer.json

With every Laravel application there is a main composer.json file in the root of every new app. This is your main application composer file and this is where we define all our app dependencies.

Let's go ahead and autoload our newly created package via the `PSR-4` autoload block:

```
"autoload": {
        "psr-4": {
            "W3public\\Calculator\\": "packages/calculator/src/"
        }
    },
```

Then we need composer to run the autoloader and autoload our package. To do this we run the following command:

```
composer dump-autoload
```

and then to optimise the class loader use the following command (not allways needed)

```
php artisan optimize
```

4. Add our package service provider.

The service provider is the main entry inside of our package. This is where our package is loaded or booted. In the root of our app, let's create our ServiceProvider with an artisan command via the command line:

```
php artisan make:provider CalculatorServiceProvider
```

This will create a new file located at `app/Providers/CalculatorServiceProvider.php`

Let's move this file into our package folder so that way it lives at `packages /calculator/src/CalculatorServiceProvider.php`

Don't forget to change your namespace to be your `Vendor_name\Package_name`.

For Example: namespace W3public\Calculator;

As you can see in our new Service Provider class we have 2 methods boot() and register(). The boot() method is used to boot any routes, event listeners, or any other functionality you want to add to your package. The register() method is used to bind any classes or functionality into the app container. We'll learn more about these in the next step.

Next, we need to add our new Service Provider in our config/app.php at the bottom of the `providers[]` array like this:

```
'providers' => [
W3public\Calculator\CalculatorServiceProvider::class,
],
```

Awesome! Our service provider is loaded and our package is ready to go! But we don't have any functionality yet... Let's tackle that by adding a routes file for our package.

5. Add our package route file

Let's start by creating a new `routes/web.php` inside of our package src directory, and add the following code:

```
<?php
Route::get('calculator', function(){
    echo 'Hello from the calculator package!';
});
```

Then let's include our web.php file inside the boot() method of our Service Provider:

```
public function boot()
{
// include __DIR__.'/routes/web.php';
$this->loadRoutesFrom(__DIR__.'/routes/web.php');
}
```

And now if we were to load up our laravel app and visit ` http://localhost:8000/calculator` route. Then we should end up with the following page:

How awesome is that! We just called a route loaded from our packge! Instead of outputting data inside of a route closure let's create a new controller and reference those methods.

6. Creating our package controller.

Let's create a new controller by running the following artisan command:

```
php artisan make:controller CalculatorController
```

Then let's move that controller from app/Http/Controllers/CalculatorController.php to packages /calculator/src/ Http/CalculatorController.php.

Next, I will create 2 methods add() and subtract() inside of this controller so the file contents will look like the following:

```
<?php
namespace W3public\Calculator;
use Illuminate\Http\Request;
use App\Http\Requests;
use App\Http\Controllers\Controller;

class CalculatorController extends Controller
{
public function add($a, $b)
{
echo $a + $b;
}

public function subtract($a, $b)
{
echo $a — $b;
}
}
```

Again, remember to change your namespace in your controller above to `Vendor_name\Package_name`;

Finally, let's add a few more routes to our `routes/web.php` file:

```
Route::get('add/{a}/{b}', 'W3public\Calculator\Http\
Controllers\CalculatorController@add');

Route::get('subtract/{a}/{b}', 'W3public\Calculator\Http\
Controllers\CalculatorController@subtract');

//OR
```

```
Route::group(['namespace'=>'W3public\Calculator\Http\Controllers
'],function(){
Route::get('add/{a}/{b}',CalculatorController@add');
Route::get('add/{a}/{b}',CalculatorController@subtract');
});
```

Then if we navigate to `http://localhost:8000/add/5/2` and `http://localhost:8000/subtract/5/2` we will end up with the following results!

Next, what if we wanted to load a few views from our package. Well, that's easy enough. Let's do that in the next step.

7. Adding views for our package.

Let's create a new folder inside of our package src folder called views. Then let's create a new file and call it `add.blade.php`. Inside the `add.blade.php` let's add the following:

```
<!DOCTYPE html>
<html>
<head>
<title>Calculator</title>
</head>
<body>
<h1 style="text-align:center">
Your Result
<span style="font-weight:normal">{{ $result }}</span>
</h1>
</body>
</html>
```

Next let's register where these views should be loaded from by adding a line of code to the boot method of our Service Provider:

```
public function boot()
{
```

```
$this->loadViewsFrom(__DIR__.'/views', 'calculator');
}
```

Lastly, let's change the add() and subtract() functions inside of our Calculator controller to call this view:

```
public function add($a, $b){
$result = $a + $b;
return view('calculator::index', compact('result'));
}

public function subtract($a, $b){
$result = $a + $b;
return view('calculator::index', compact('result'));
}
```

If we load up the following URL http://localhost:8000/add/5/2 in our browser we'll end up with the following:

Finally you have done it!

That's how to create a Laravel package. If you want you can also do more about how to publish your views and a few other advanced topics on publishing your package. Now let's do this

Publishing Views

If we want to change markup or any other things we have to publish views, for publishing to the application's resources/views/vendor directory, we may use the service provider's **publishes** method. The publishes method accepts an array of package view paths and their desired publish locations like this

```
public function boot()
{
   $this->loadViewsFrom(__DIR__.'/ views', calculator');

   $this->publishes([
      __DIR__.'/views' =>
resource_path('views/vendor/calculator'),
   ]);
}
```

Now run the **vendor:publish** Artisan command then the calculator.blade.php file will be copied to the specified publish location.

Submitting the package to the packagist.org

If you want to submit it to to packagist directory so that way you can install your packages with composer you'll need to do the following steps:

- Firstly we need Package Discovery for automatically add the service provider to the providers array. So add the following things to the packages/composer.json file

```
"extra": {
   "laravel": {
      "providers": [
         "W3public\\Calculator\\CalculatorServiceProvider"
      ]
```

```
    }

},
```

- Submit your package to Github (download form https://github.com/w3public/calculator)
- Submit your package and Github URL to Packagist (install form here https://packagist.org/packages/w3public/calculator)

And then you'll be able to include your package inside of your main project composer.json file and run composer install/require to install your package in your application.