Laravelتوسط سر تیلور اتول به عنوان تلاشی برای جایگزینی عالی برای چارچوب قدیمی‌تر PHPبا نام CodeIgniter ایجاد شد. دلیل این امر این بود که CodeIgniter ویژگی‌های فوق العاده‌ای مانند پشتیبانی از احراز هویت داخلی مشتری و مجوز مناسب کاربر را ارائه نکرده است.

در نهم ژوئیه ۲۰۱۱، Laravelاولین نسخه بتا خود را منتشر کرد و بعداً در همان ماه، Laravel 1 منتشر شد. به غیر از تأیید هویت، لاراول همچنین از پشتیبانی محلی، نمایش‌ها، مدیریت نشست‌ها، مسیریابی، درخواست به کنترل کننده خاص و سایر ویژگی‌های شگفت انگیز برخوردار است.

این فریم ورک پر از کتاب خانه های مختلف است که به برنامه نویسان تازه کار و حتی حرفه ای کمک میکند.بر پایه معماری MVCمی باشد. با استفاده از لاراول، تیلور در صدد ایجاد چارچوبی بود که به دلیل سادگی شناخته شود .لاراول دارای امکانات و ویژگی های به خصوصی است که بر سرعت برنامه تاثیر می گذارد. این ویژگی ها عبارت اند از:

Service container

Queues

Events

APP

پوشه APP پوشه اص لی برنامه می باشد. و تقریبا تمام فایل های اصلی و هسته برنامه داخل این پوشه میباشد. بطور پیشفرض namespace ی که لاراول برای این دایرکتوری قرار داده است APP میباشد

Console

شامل تمام دستورات آرتیسان مورد نیاز لاراول است.

Http

پوشه http شامل پوشه های کنترلر ها و میلدور ها و در خواست های اپلیکشن است.

Bootstrap

این پوشه شامل تمام فایل های راه انداز برنامه می باشد. این دایرکتوری شامل پوشه cacheاست و شامل تمام فایل های لازم برای کش کردن یک برنامه وب می باشد .

config

این پوشه شامل تمام فایل های پیکر بندی و تنظیمات و پارامترها ی مربوط به کارکرد صحیح یک برنامه لاراول می باشد

database

همانطور که از آن پیداست این دایرکتوری شامل پارامترهای مختلف برای مدیریت پایگاه داده است

public

این پوشه شامل فایل ها و پوشه های زیر است:

.htaccess**بوسیله این فایل یکسری تنظی مات به سرورمون اعمال میکنیم**.

– javascript and cssدر **دایرکتوری** public **همچنین** asset **های برنامه مانند** javascript**و یا** css **قرار می گیرد**.

– index.php**ای ن فایل برای راه اندازی برنامه ، لازم و حیاتی و اولین نقطه برای درخواستهای وارد شده به برنامه است**.

resources

این دایرکتوری برای بهبود هر چه بیشتر برنامه وب شما می باشد.

storage

در این دایرکتوری فای ل هایی که در جریان پروژه توسط لاراول کامپایل یا ساخته میشوند قرار میگی رد. مثلا فایلهای مربوط به cache **یا** log **های برنامه و یا قالب** blade**که کامپایل شده است** .

tests

در این دایرکتوری فایلهایی برای تست برنامه گنجانده میشود. مثلا ما می توانیم بطور پیش فرض با PhpUnit **کار کنیم . فایلهای موردنظر برای تست را در این پوشه قرار می دهیم**.

vendor

این دایرکتوری تمام dependency **ها یا وابستگی ها و پکیج های مربوط به کامپوزر را در بر می گیرد**.

سوال ۲

Artisan یک ابزار خط فرمان در فریم‌ورک لاراول است.
این ابزار وظیفه دارد تا انجام تغییرات در فریم‌ورک را ساده کند. فراخوانی این ابزار با استفاده از فایل اجرایی php به انجام می‌رسد
برخی از کارهایی که **Artisan** انجام می‌دهد عبارتند از:

فعال و غیر فعال کردن سایت
بهینه‌سازی فریم‌ورک برای عملکرد بهتر
به‌روزرسانی پایگاه داده با آخرین تغییرات
ایجاد کلید امنیتی برنامه که در کوکی‌ها و سشن‌ها استفاده می‌شود

برای فراخوانی دستورات **artisan** از خط فرمان به شکل زیر عمل می‌کنیم

```php
php artisan [options] command [arguments]
```

مشاهده‌ی دستورات قابل استفاده در **artisan** با نوشتن دستور زیر امکان‌پذیر است
کد:

```php
php artisan
```

برای دیدن لیست کامل کامنت های **artisan** از دستور زیر استفاده کنید

```php
php artisan list
```

هر کدام از دستورهای artisan یک راهنما دارد برای دیدن صفحه راهنمای آن می توانید مانند زیر عمل نمایید.

```php
php artisan help migrate
```

برای نمایش ورژن لاراول نصب شده روی سیستمون میتونید از دستور زیر استفاده نمایید.

```php
php artisan --version
```

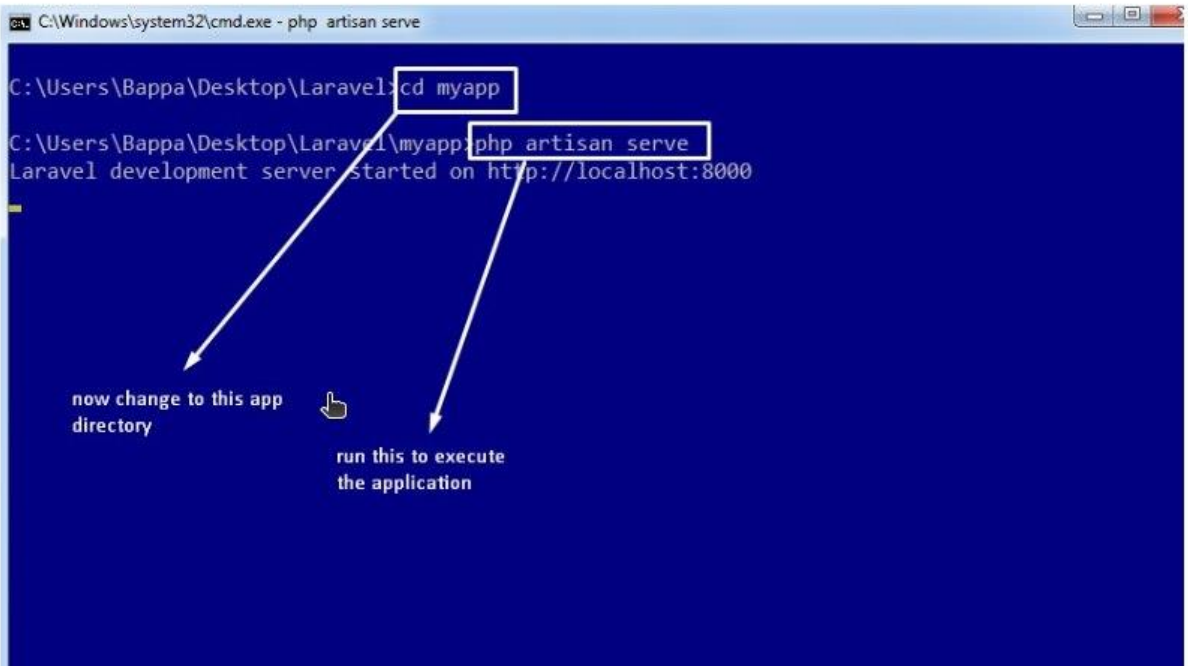
یکی از پرکاربردترین دستورات artisan دستور dump-autoload است
کد:

```php
php artisan dump-autoload
```

وظیفه‌ی این دستور، جستجوی تمام شاخه‌های برنامه و ایجاد فایل autoloader است. پس از اینکه یک کلاس به سایت اضافه کنید یا چیزی از آن را حذف کنید، لازم است تا این دستور صدا زده شود.
فایل autoloader حاوی فهرست تمامی کلاس‌ها و آدرس قرارگیری آن‌هاست. با در اختیار داشتن یک فایل autoloader به‌روز و مرتب‌شده، سرعت کلی عملکرد سایت افزایش پیدا می‌کند.

با دستور زیر می توانید بدون نیاز به فعال کردن xampp یا wamp سایت خود را به صورت لوکال در مرورگر تست کنید.

```php
php artisan serve
```



۳)

View یکی از بخش‌های سه گانه **MVC** است که در این قسمت قصد داریم به آن بپردازیم. همانطور که در قسمت اول از آموزش لاراول اشاره کردیم، این فریم‌ورک از **تمپلیت انجین قدرتمند Blade** بهره می‌برد.

Template Engine یا موتور قالب یک سیستم حرفه‌ای برای مدیریت قالب برنامه‌هایی است که می‌نویسیم. با یک موتور قالب می‌توانید راحت تر قالب‌های خود را برنامه نویسی کرده و از مزایای متعددی بهره مند شوید.

Template Engine استفاده شده در لاراول، Blade نام دارد که از قدرت بالایی برخوردار است. این موتور قالب حرفه‌ای مزایای متنوعی دارد که در اینجا به چند مورد اشاره می‌کنیم:

- اگر از Blade استفاده کنید **امنیت** را به سایت خود هدیه می‌دهید، چون مقادیر و متغیرها پس از بررسی‌های امنیتی در سایت قرار می‌گیرند.
- می توانید خیلی راحت تر و تمیز تر از PHP **پوسته گرافیکی** خود را تولید کنید.
- استفاده از Blade منجر به **کش شدن** بهتر صفحات خواهد شد.
- استفاده از Blade **عیب یابی** از کدهای شما را تسهیل می‌بخشد.
- کار با Blade و یاد گرفتن آن بسیار آسان است.
- اگر از Blade استفاده کنید می‌توانید همزمان از **کدهای PHP** درون قالب خود استفاده کنید.

در مثال زیر Controller متغیر `$name` را برای فایل پوسته با نام `show.blade.php` ارسال می‌کند:

```
1  public function show()
2  {
3      return view('show', ['name' => 'hitos']);
4  }
```

در پایین کدی را می‌بینید که در آن متغیر `$name` به نمایش در می‌آید:

```
1  {{$name}}
```

همانطور که در بالا و در مزایای Blade گفتیم که شما در این موتور قالب می‌توانید از کدهای PHP نیز استفاده کنید. بنابراین اگر در `show.blade.php` از کد `echo $name;` استفاده کنیم همان خروجی قبل را دریافت می‌کنیم.

در پایین کدی را می‌بینید که در آن متغیر `$name` به نمایش در می‌آید:

```
{{$name}}
```

همانطور که در بالا و در مزایای Blade گفتیم که شما در این موتور قالب می‌توانید از کدهای PHP نیز استفاده کنید. بنابراین اگر در `show.blade.php` از کد `echo $name;` استفاده کنیم همان خروجی قبل را دریافت می‌کنیم.

برای کامنت کردن بخشی از قالب‌های لاراول، سورس کد خود را بین `{{--` و `--}}` قرار دهید:

```
{{-- کد این بخش توسط موتور قالب نادیده گرفته می‌شود --}}
```

در قالب‌های پیشرفته باید بتوانیم درون قالب متغیرهایی را بوجود آورده و یا ویرایش کنیم. برای تعریف یک متغیر جدید دستورات را بین `{{--*/` و `/*--}}` قرار می‌دهیم. مانند:

```
{{--*/ $variable = 0 /*--}}
```

حال متغیر فوق را به سادگی و در هر جایی از قالب به صورت `{{ $variable }}` استفاده می‌کنیم.

استفاده از حلقه‌ها در قالب‌های Blade و لاراول

در پایین قصد داریم استفاده از حلقه‌های زیر را به شما آموزش دهیم:

- foreach: تکرار تا تکمیل تمام عناصر
- for: تکرار به تعدادی معین و با وجود شرط
- while: تکرار تا جایی که شرط خاصی برقرار باشد

استفاده از حلقه‌ها در قالب‌های Blade و لاراول

در پایین قصد داریم استفاده از حلقه‌های زیر را به شما آموزش دهیم:

- foreach: تکرار تا تکمیل تمام عناصر
- for: تکرار به تعدادی معین و با وجود شرط
- while: تکرار تا جایی که شرط خاصی برقرار باشد

آموزش استفاده از دستور Foreach

دستور Foreach برای نمایش آرایه‌ها بسیار کاربردی و ضروری است. فرض کنید در Controller یک تابع با دستورات زیر دارید:

```
class MainController extends Controller
{
    public function index()
    {
        $names= ["Ali", "Hasan", "Alireza", "Mohsen"];
        return view('show', ['names' => $names]);
    }
}
```

فایل show.blade.php به شکل زیر است:

```
<ul>
@foreach($names as $name)
    <li>{{ $name }}</li>
@endforeach
</ul>
```

خروجی این فایل را در پایین می‌بینید:

۴)

چرخه ی حیات درخواست (Request Lifecycle)

1. مقدمه
2. مرور کلی بر چرخه ی حیات
3. شرح دقیق Service Provider

مقدمه

به هنگام استفاده از هر ابزاری در دنیای واقعی، مسلما آشنایی با نحوه ی استفاده از آن اطمینان خاطر بیشتری به شما می دهد. برنامه نویسی هم از این قاعده مستثنی نیست. زمانی که می دانید چگونه ابزار توسعه ی شما کار می کنند قطعا استفاده از آن ها برای شما آسان تر می شود.

هدف اصلی این آموزش ارائه ی مرورکلی و سطح بالا بر نحوه ی عملکرد فریم ورک لاراول می باشد. پس از آشنایی با چارچوب کلی این فریم ورک، استفاده از همه چیز نسبتا آسان تر می شود و متعاقبا برنامه ی کاربردی خود را با اعتماد به نفس بیشتری طراحی خواهید کرد.

اگر با تمامی اصطلاحات موجود در این برنامه آشنایی ندارید، جای هیچ نگرانی نیست. فقط کافی است دانش ابتدایی از آنچه رخ می دهد بدست آورید. خواهید دید که با خواندن بخش های مختلف این آموزش دانش شما افزایش یافته و درک بهتری از مفاهیم آن پیدا خواهید کرد.

مرور کلی بر چرخه ی حیات

Entry point (نقطه ی شروع اجرای برنامه) تمامی درخواست ها به اپلیکیشن تحت وب لاراول فایل public/index.php می باشد. تمامی درخواست ها بر اساس پیکربندی سرور مجازی (وب سرور Apache / Nginx) به این فایل هدایت می شوند. فایل index.php حاوی مقدار کد زیادی نیست. در واقع فایل نام برده صرفا یک نقطه ی شروع برای بارگذاری باقی چارچوب نرم افزاری می باشد.

فایل index.php تعریف autoloader که توسط Composer تولید شده را بارگذاری می کند و سپس یک نمونه از برنامه ی کاربردی Laravel را از اسکریپتbootstrap/app.php بازیابی می کند. اولین عملیاتی که خود لاراول انجام می دهد ایجاد یک نمونه از برنامه ی کاربردی / Service container می باشد.

هسته های HTTP / console

سپس درخواست ورودی، بسته به نوع آن، یا به هسته ی HTTP و یا به هسته ی console ارسال می گردد. این دو هسته (kernel) به عنوان یک نقطه ی مرکزی برای تمامی درخواست های ورودی ایفای نقش می کند. بدین معنی که تمامی درخواست ها از این دو هسته عبور می کنند. اجالتا فقط به شرح هسته ی HTTP مقیم در مسیر app/Http/Kernel.php می پردازیم.

هسته ی HTTP از کلاس Illuminate\Foundation\Http\Kernel ارث بری می کند. این کلاس آرایه ای از bootstrapper ها تعریف کرده که پیش از اجرای درخواست اجرا می شوند. این bootstrapper ها (خود راه اندازه ها) وظیفه ی پیکربندی مدیریت خطاها، ثبت وقایع (logging)، همچنین تشخیص محیط برنامه و دیگر کارهایی که بایستی پیش از اداره و اجرای (خود) درخواست اجرا شوند را بر عهده دارد.

هسته ی HTTP همچنین یک لیست از middleware ها را تعریف می کند که تمامی درخواست ها بایستی پیش از مدیریت شدن توسط برنامه ی کاربردی از آن عبور کنند. این middleware ها وظیفه ی خواندن و نوشتن HTTP session، بررسی اینکه آیا برنامه در حالت تعمیر و نگهداشت قرار دارد یا خیر (maintenance mode) و نیز بررسی توکن CSRF را برعهده دارند (middleware یک سازوکار بهینه برای فیلتر کردن درخواست های HTTP تعبیه می کند. به عنوان مثال می توان به middleware تصدیق هویت برای login و ثبت ورود کاربر اشاره کرد).

امضای متد handle، از توابع هسته ی HTTP، بسیار ساده و قابل فهم می باشد: یک Request دریافت کرده و یک Response برگرداند (منظور از امضا یا signature متد همان اسم متد و تعداد پارامترهای ورودی و نیز نوع آن ها می باشد). می توانید به Kernel یا هسته به چشم یک جعبه ی سیاه نگاه کنید که کل برنامه ی شما را تشکیل می دهد (بیانگر کل برنامه ی شما می باشد). درخواست های HTTP را به آن خورانده و در خروجی پاسخ های HTTP را دریافت نمایید.

Service Provider

یکی از مهمترین عملیات (تنظیم و راه اندازی به صورت خودکار) bootstrapping که هسته (kernel) عهده دار آن است، بارگذاری service provider برای اپلیکیشن تحت وب شما می باشد (منظور از bootstrapping ثبت و رجیستر binding و اتصال event listener، service container و middleware و حتی route ها است). تمامی service provider های برنامه در فایل پیکربندی config/app.php متعلق به آرایه یproviders تنظیم می شوند. به این صورت است که ابتدا متد register بر روی تمامی provider ها فراخوانی می شود، بعد از اینکه تمامیprovider ها ثبت و ایجاد شدند، آنگاه متد boot صدا زده می شود.

گفتنی است که service provider ها مسئولیت تنظیم و راه اندازی خودکار (bootstrapping) اجزاء مختلف برنامه را بر عهده دارند که از جمله یآن ها می توان به کامپوننت های validation، queue، database و routing اشاره کرد. از آنجایی که service provider ها تمامی ابزار و امکانات ارائه شده توسط فریم ورک لاراول را به صورت خودکار راه اندازی (bootstrap) و پیکربندی می کنند، می توان گفت که (service provider) مهمترین بخش کل فرایند تنظیم و راه اندازی خودکار لاراول محسوب می شوند.

ارسال درخواست (Dispatch Request)

پس از اینکه برنامه به صورت خودکار راه اندازی (bootstrap) شده و همچنین تمامی service provider ها کاملا ثبت شدند، Request برای ارسال (Dispatch) به router واگذار می شود. router علاوه بر ارسال درخواست به route یا controller، تمامی middleware های مختص بهroute نیز را اجرا می کند.

شرح دقیق Service Provider

Service provider ها مرکز پیکربندی و انجام تنظیمات پایه ای برنامه می باشند. به این صورت است که ابتدا نمونه ی اپلیکیشن ایجاد می شود و به دنبال آن service provider ها ثبت گردیده و سپس درخواست به اپلیکیشن تنظیم شده (bootstrapped app) فرستاده می شود. به همین راحتی!

داشتن فهمی عمیق از چگونگی ساخته شدن برنامه و تنظیم آن توسط service provider بسیار ازشمند می باشد. لازم به ذکر است که providerهای پیش فرض اپلیکیشن در پوشه ی app/Providers نگهداری می شوند.

به طور پیش فرض، AppServiceProvider تقریبا تهی می باشد. این provider مکان یا ظرف بسیار مناسبی برای افزودن service container binding ها و نیز فرایند پیکربندی (bootstrapping) اختصاصی برنامه ی تحت وب شما می باشد. لازم به گفتن نیست که برای برنامه های بزرگ بایستی چندین service provider ایجاد کرده و برای هر یک bootstrapping با درجه ی granuality بیشتر درنظر بگیرید (granularity به تعیین سطح جزیئات سیستم اشاره دارد).

(5

:

وقتی میخوای یه کنترلر بسازی ، اگه آخرش r- **یا** resource-- **روبزنی کنترلر به همراه چند متود داخلش ایجاد میشه**

Route**های** resource **کلا متدهای** restful **رو در خودشون جا دادند که شامل متدهای** store, index, show, update, destroy, edit, create **میشه**

هدف از این مفهوم خلاصه و کم حجم کردن روتهای هست که تعریف کردیم و به راحتی متوانید با قطع کد زیر روت های بالا را مدیریت و خلاصه کنید:

```
php artisan make:controller BookController –resource
```

(6

هنگامی روت ها درون برنامه ما ایجاد می شوند ممکن است تعدادشان خیلی زیاد شود و اگر بخواهیم از آنها برا ی مسی ردهی های داخل برنامه ی خودت استفاده کنیم یادآوری و به کار بردن آنها نسبتا سخت میشه و همچنین ممکن بعدا عوض بشوند، برای همین مفهوم Named **به ما کمک می کند تا برا ی روت های خود یک اسم کوتاه و مختصر بگذاریم**

مثال:

```
Route::get('/posts/laravel-project-10221/nz7eX') ->name('post')
```

(۷

اگر دیتا را از کنترلر به ویو با روش زیر

```
$first = "<b>Narendra Sisodia</b>";
```

منتقل کنیم و در blade **با** {{$first}} **به آن دسترسی پیدا کنیم به شکل زیر خواهد بود**

```
< b>Narendra Sisodia</b >
```

اما اگر از این }!! $ first !!{ **استفاده کنیم خروجی به این شکل خواهد بود**:

Sisodia Narendra