 از لاراول می‌توان به عنوان محبوب‌ترین فریم ورک زبان برنامه نویسی  یاد کرد. این فریم ورک توانسته است در زمانی کوتاه، به اولین گزینه‌ی توسعه دهندگان برای توسعه برنامه‌های پیچیده تحت وب تبدیل شود.

 لاراول فریم ورکی مبتنی بر معماری **MVC** است که به صورت رایگان و متن باز ارائه شده است. این فریم‌ ورک امکانات هوشمندی در اختیار برنامه نویسان قرار می‌دهد و برنامه‌‌ نویسی را برای آن‌ها لذت بخش‌تر می‌کن

این فریم ورک در بین سایر فریم ورک‌های PHP از امنیت و سرعت توسعه‌ی بیشتری برخوردار است و از طریق آن شما قادر خواهید بود بدون اینکه تغییری در کدهای اصلی ایجاد کنید، عملکرد بسیاری از بخش‌ها را تغییر دهید. لاراول کتابخانه‌های بسیار متنوعی دارد و از اغلب کتابخانه‌های فریم ورک سیمفونی نیز استفاده می‌کند. همچنین این فریم ورک سرویس‌های مختلفی را برای برنامه نویسان ارائه می‌دهد. به عنوان مثال سرویس هایی همچون تأیید هویت، دسترسی‌های کاربران، کنسول Artisan، پرداخت و صورتحساب، کش (Cache) و … که تنها بخش کوچکی از سرویس هایی است که این فریم ورک در اختیار برنامه‌ نویسان قرار می‌دهد.

- app

-- console

پوشه ی console شامل دستورات Artisan فریمورک است که ابزاری مهم برای لاراول است.درون پوشه ی console یک پوشه ی دیگری به نام Commands وجود دارد که محل دستورات artisan است.

فایل Kernal.php دستورات درون فایل Inspire.php را صدا میزند

-- Exceptions

این فولدر شامل تمامی متود ها برای کنترل کردن استثناها یا exception در لاراول است.درون این دایرکتوری یک فایل به نام handle.php وجود دارد که تمام این کارها را انجام میدهد و exception ها را مدیریت و کنترل میکند.

-- Http

این دایرکتوری یکی از مهمترین بخش های فریمورک لاراول است. در این دایرکتوری چند زیردایرکتوری به نام های controllers ، middleware و application requests موجود است.

از آنجایی که لاراول از الگوی MVC پیروی میکند این دایرکتوری شامل model ، controller و view های تعریف شده برای هربخش میباشد.

زیردایرکتوری دیگری به نام Middleware وجود دارد که مکانیسم middleware لاراول درآنجا تعریف میشود که فیلترهای مختلف و مکانیسم های لازم برای مدیریت درخواست های بین request و respons را کنترل میکند.

زیردایرکتوری Requests شامل تمامی request های مربوط به اپلیکیشن لاراولی است.

Models--

جایی هست که با دیتابیس ما درگیر میشود

-- Providers

این دایرکتوری شامل service provider ها میباشد که برای رجیستر کردن event ها برای سرورهای اصلی میباشد و همچنین اپلیکیشن لاراول را کانفیگ میکند.

**- Bootstrap**

این فولدر شامل تمام اسکریپت های Bootstrap اپلیکیشن  است. این فولدر دارای یک زیردایرکتوری به نام cache میباشد که شامل تمام فایل های مربوط به کش اپلیکیشن لاراولی است.

همچنین شما میتوانید یک فایل به نام app.php را پیدا کنید که راه اندازی اولیه اپ است.

**- Config**

فولدر config شامل تنظیمات و پارامترهای مرتبط مختلف است تا[اپلیکیشن](https://avasam.ir/product/23" \o "دوره ی آموزش فریمورک لاراول" \t "_blank)  عملکرد صحیح خود را داشته باشد.فایل های زیادی در این بخش وجود دارد نام فایل ها نشان دهنده ی عملیات آن فایل است.

**- Database**

همانطور که از نام این فولدر پیداست این فولدر شامل پارامتر های مختلف برای کار با پایگاه داده است.این بخش شامل سه زیردایرکتوری است که در زیر آمده است.

 -- Seeds -> این بخش برای ساخت اطلاعات تستی یا آزمایشی در پایگاه داده استفاده میشود.

-- Migrations -> این بخش شامل تمام مایگریشن هاست که عمل کوئری روی پایگاه داده انجام میدهند.

-- Factories -> این بخش برای تولید داده ها با حجم زیاد استفاده میشود.

**Public**

این فولدر همانطور که از نام آن پیداست بخش عمومی و برای پیاده سازی اولیه استفاده میشود. این بخش شامل فایل ها و فولدرهای زیر میشود.

 -- .htaccess این فایل برای تنضیمات مورد استفاده قرار میگیرد.

-- javascript و css این ها بعنوان assets تعریف شده اند.

-- index.php این اولین فایلی است که از اپلیکییشن اجرا میشود و برای اجرا اولیه اپلیکیشن لاراولی مورد نیاز است.

**Resources**

فولدر Resources دارای فایل هایی هست که اپلیکیشن لاراولی شما را تقویت میکند.زیردایرکتوری های این بخش در زیر به همراه جزيیات آنها آورده شده است.

-- assets این دایرکتوری شامل فایل های LESS و SCSS است که برای استایل دهی به سایت لاراولی مورد نیاز است.

-- lang این بخش برای تنظیمات محلی و زبان است که برای منطقه ی مورد استفاده ی لاراول قابل تنظیم است.

-- views ویو ها فایل های HTML یا تمپلیت هستند که قالب اصلی سایت را برای مشاهده توسط بازدید کننده فراهم میکنند . این بخش جزو یکی از بخش های MVC است.

.

routes

که 4 فایل قرار دارد

-> که مسیر هایی که میسازیم قرار داردweb

-> مسیر ها برای ارسال دربافت ای پیاapi

-> برای کامند های کنسول console

**Storage**

این پوشه ای است که همه logs و فایل های لازم را که اغلب در هنگام اجرای پروژه Laravel مورد نیاز است، ذخیره می کند.

زیرپوشه های موجود در این پوشه و هدف آنها در زیر آمده است.

app : این پوشه حاوی فایل هایی است که به صورت متوالی نامیده می شوند.

framework : شامل جلسات، cache و views است که frequently نامیده می شوند.

logs :  همه استثناء ها و log ها در این زیر پوشه ردیابی می شوند.

**- Tests**

تمام واحد های تست در این بخش قرار داده میشوند.سیستم نامگذاری camel\_case است یعنی تمام کلمات به هم چسبیده هستند و حرف اول همه ی کلمات با حرف بزرگ نوشته میشود ولی حرف اول کلمه ی اول با حرف کوچک نوشته میشود به این صورت thisIsPhpFrameworkForYou ...

**- Vendor**

فریمورک لاراول کاملا مبتنی بر وابستگی های composer است برای مثال هنگام نصب لاراول یا تنظیم یا نصب یک کتابخانه ی جانبی شما از composer بهره میگیرید. فولدر Vendor شامل تمام وابستگی های ( dependencis ) فریمورک لاراول است.

علاوه بر پرونده های ذکر شده در بالا ، لاراول شامل فایلهای دیگری نیز می باشد که در عملکردهای مختلف مانند پیکربندی GitHub ، بسته ها و کتابخانه های جانبی نقش اصلی را ایفا می کنن

2

artisan.

Artisan یک رابط خط فرمان (CLI) اختصاصی لاراول هست که شامل دستورات کاربردی مثل ساختن کنترلرها، مایگریشن‌ها، میدلویرها و چندین دستور دیگه میشه و کار رو برای توسعه‌دهنده‌ها خیلی راحت‌تر و سریع‌تر کرده. اما یک قابلیت جالب‌تری وجود داره اینه که ما هم میتونیم دستورات اختصاصی و دلخواه خودمون رو بنویسیم و از اون توی پروژه یا پکیجمون استفاده کنیم

php artisan make:command UserExists

.

3

= template engines

در واقع رابط بین مقدارها یا متغیرهای برنامه های سمت سرور (php) با HTML می باشد و دیگر نیازی نیست در صفحه های HTML سایتمون از کدهای PHP استفاده کنیم و کدنویسی کثیفی رو ایجاد کنیم.

پس دیگر نیازی نیست مثلا بنویسیم echo titr  و به جای ان فقط می نویسیم {titr}.

.این یکی از قوانین تمپلت انجین ها هست .حالا نحوه کار هر تمپلت انجینی کمی با بقیه فرق میکنه و هر کدوم ویژگی های خاص خودشون را دارن. ولی قایده کلی آن ها به این نحو هست که عرض کردم.

برای حلقه های for  ویا  دستور شرطی if و … هم ساختارهای خوبی وجود دارد.

Blade موتور قالب ساده و در عین حال قدرتمندی است که در لاراول گنجانده شده است. برخلاف برخی از موتورهای قالب PHP، Blade شما را از استفاده از کد PHP ساده در قالب های خود محدود نمی کند. در واقع، همه قالب‌های Blade در کدهای PHP ساده کامپایل می‌شوند و تا زمانی که اصلاح شوند در حافظه پنهان ذخیره می‌شوند، به این معنی که Blade اساساً سربار صفر به برنامه شما اضافه می‌کند. فایل های قالب Blade از پسوند فایل .blade.php استفاده می کنند و معمولاً در منابع ذخیره می شوند

4

**مرحله اول : Auto Loader**

به عنوان اولین مرحله ، درخواست از مرورگر کاربر شروع می شود ، سپس به سرور وب می رسد. سرور وب (Apache یا Nginx) درخواست داده شده را به فایلی در مسیر Laravel public / index.php هدایت می کند که یک نقطه شروع برای بارگذاری بقیه فریمورک است. این فایل autoloader ای که توسط composer ساخته شده است را لود میکند.

سپس با لود شدن autolader ، نمونه ای از برنامه Laravel از اسکریپت bootstrap / app.php بازیابی میشود. یعنی در اولین مرحله ، لاراول خودش نمونه ای از برنامه را ایجاد می کند. بطور مثال بوت استرپ را در این مرحله میشود اینگونه تشبیه کرد که وقتی بعنوان راننده سوار میشید، اولین مرحله چک کردن آینه ها و بستن کمربند و .... میباشد.

**مرحله دوم: Kernel**

مرحله بعدی در قسمت هسته برنامه( KERNEL) اعمال خواهد شد.

بسته به نوع درخواستی که وارد برنامه می شود ، درخواست دریافتی به HTTP kernel یا console kernel ارسال می شود.

این دو نوع KERNEL به عنوان مکان اصلی برای اجرای همه درخواست ها میباشند.

هسته یا کرنل HTTP ، در برنامه Http / Kernel.php قرار داده شده است. این kernel فقط یک درخواست دریافت می کند و یک پاسخ ارسال می کند. بوت استرپرهایی که توسط کلاس Kernel تعریف شده اند ، مسئولیت رسیدگی به خطا ، پیکربندی ورود به سیستم (configure logging) ، مسیریابی های تعریف شده در فایل env و سایر کارهایی را که باید قبل از انجام درخواست ارسالی انجام شود را برعهده دارد .

کرنل HTTP همچنین لیستی از middleware هایی که قبل از استفاده از برنامه باید مورد توجه قرار بگیرد را نیز تعریف خواهد کرد.

**مرحله سوم: Service Providers**

مرحله بعداز گذشتن از kernelها، لود کردن ارائه دهندگان خدمات (Service Providers) هستند که بخشی از عملکرد های bootstrapping میباشند. Service Providers مورد نیاز برای برنامه در فایل onfig/app.php قرار دارند.

همینطور که متدهای تعریف شده صدا زده میشوند، تمام providers ثبت می شوند. پس از ثبت و تعریف همه ی providers ، روش های بوت providers ها صدا زده میشود. providers ها روش های بت متفاوتی دارند که در با استفاده از آنها صدا زده میشود که در مقالات آتی به آنها میپردازم.

**مرحله چهارم: Dispatch Request**

پس از راه اندازی برنامه و ثبت و راه اندازی همه Service Providers، درخواست برای ارسال به روتر تحویل داده می شود. که همان web.php یا app.php میباشد. سپس روتر درخواست را به یک مسیر یا کنترلر مدنظر ارسال می کند ، و همچنین در صورت نیاز هر middleware خاصی را برای مسیر اجرا می کند.

**مرحله پنجم: Router**

اکنون درخواست توسط روتر به کنترلر یا مسیر موردنظر منتقل می شود و ریکوئست ارسالی مطابق روند زیر در پایان قابل مشاهده خواهدبود:

روتر درخواست های HTTP را به کنترلر هدایت می کند یا با حذف کنترلر ، نمایش یا پاسخ ها را مستقیماً باز می گرداند. این مسیرها در فایل app/routes.php قرار دارند.

کنترلر ها که در مسیر app/controllers/ قرار دارند، اقدامات خاصی را انجام داده و داده ها را به یک view ارسال می کند.

ویوها که در مسیر app/views/ قرار دارد، قالب داده ها را بصورت مناسبی جهت پاسخ در HTTP آماده میکند و ارائه میدهد

**6و5**

**روت Resource و Named چیست و چه کاربردی دارد ؟**

هنگامی روت ها درون برنامه ما ایجاد می شوند ممکن است تعدادشان خیلی زیاد شود و اگر بخواهیم از آنها برای مسیردهی 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 برای روت های خود یک اسم کوتاه و مختصر بگذاریم به کد زیر دقت کنید :

**Route::get('/posts/laravel-project-10221/nz7eX', '...')->name('post');**

حالا بجای نوشتن همچین ادرس سختی کافی است فقط نام آن را صدا بزنیم در صفحه های **html** خود به شکل زیر :

**<form action="{{ route('post') }}"></form>**

خب مفهومی بعدی که میخواهیم در مورد آن صحبت کنیم **Resource** می باشد خیلی موقع ها شما یک عملیات تکرای مثل **CRUD** را در نوشتن روت های خود دارید یک روش این است که هر دفع بیاید برای یک موجودیت یا مدل هر چهار عمل را تکرار کنید به شکل زیر :

**Route::get('books', 'BookController@index');**

**Route::get('books/{book}', 'BookController@show');**

**Route::post('books', 'BookController@create');**

**Route::patch('books/{book}', 'BookController@update');**

**Route::delete('books/{book}', 'BookController@destroy');**

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***

در بخش روت ها به جای تمام آنها کد زیر را داریم :

**Route::resource('books', 'BookController');**

7

تفاوت اصلی آنها در {{}} برای چاپ کردن یک متغیر را چاپ میکند حال اگر متن داخلی آن حاوی تگ های html باشد آن هارو چاپ نمی کند و رندر نمیکند اگر تگ اسکریپت ساده باشد آن را به صورت زیر مینویسد

<script> alert(‘hack’) </script>

رندر نمیکند

حال اگر { !! {!! با این باشد این تگ رندر و میخواند

https://laravel.com/docs/9.x/blade