

Laravel Framework কিভাবে ইনস্টল করব?

1. xampp server : download latest version and install
2. composer: downloaod laravel composer from <https://getcomposer.org/Composer-Setup.exe> and install.

=====Laravel command=====























Click on **start** button on pc

Open commandprompt: **cmd** <--|

change directory: **cd**

1. **Composer check:** C:\xampp>Composer
2. **Create laravel Project:** C:\xampp\htdocs>composer create-project laravel/laravel your-project-name --prefer-dist
3. **login to project:** C:\xampp\htdocs>cd your-project-name
4. **to navigate/start your project:** C:\xampp\htdocs>your-project-name>php artisan serve

Default Laravel Structure:

Name	Date modified	Type	Size
 app	17-Dec-17 6:15 PM	File folder	
 bootstrap	17-Dec-17 6:15 PM	File folder	
 config	17-Dec-17 6:15 PM	File folder	
 database	17-Dec-17 6:15 PM	File folder	
 public	17-Dec-17 6:15 PM	File folder	
 resources	17-Dec-17 6:15 PM	File folder	
 routes	17-Dec-17 6:15 PM	File folder	
 storage	17-Dec-17 6:15 PM	File folder	
 tests	17-Dec-17 6:15 PM	File folder	
 vendor	17-Dec-17 6:26 PM	File folder	
 .env	17-Dec-17 6:26 PM	ENV File	1 KB
 .env.example	17-Dec-17 6:15 PM	EXAMPLE File	1 KB
 .gitattributes	17-Dec-17 6:15 PM	GITATTRIBUTES File	1 KB
 .gitignore	17-Dec-17 6:15 PM	GITIGNORE File	1 KB
 artisan	17-Dec-17 6:15 PM	File	2 KB
 composer.json	17-Dec-17 6:15 PM	JSON File	2 KB
 composer.lock	17-Dec-17 6:26 PM	LOCK File	139 KB
 package.json	17-Dec-17 6:15 PM	JSON File	2 KB
 phpunit.xml	17-Dec-17 6:15 PM	XML File	2 KB
 readme.md	17-Dec-17 6:15 PM	MD File	4 KB
 server.php	17-Dec-17 6:15 PM	PHP File	1 KB
 webpack.mix.js	17-Dec-17 6:16 PM	JS File	1 KB

- **app:**

এটি মূলত আপনার জন্য, অর্থাৎ এই ডিরেক্টরিতে আপনার এপ্লিকেশন এর Core কোড গুলো থাকবে , যেগুলো আপনিই বানাবেন । যেমনঃ Controller, Middleware সহ অন্যান্য প্রয়োজনীয় class গুলো।

- **bootstrap:**

এটির মধ্যে **app.php** ফাইলটি থাকে, যা দিয়ে Laravel Framework টি কে bootstraps করে। তা ছাড়া এই ফোল্ডার এর মধ্যে একটি cache folder ও থাকে, যার মধ্যে performance optimization এর জন্য Framework Generated file এবং folder গুলো থাকে , যেমন routes এবং services cache files .

- **config**

এই Folder এর মধ্যে আপনার application এর সব configuration ফাইল গুলো থাকবে। যেমন ডাটাবেস, মেইল, সেশন ইত্যাদি।

- **database**

এখানে আপনার application এর সব database migration এবং seeds থাকবে । এমনকি SQLite database ফাইলটাও এখানে থাকে। database migration এবং seeds কি? পরে বিস্তারিত আলোচনা করব।

- **public**

Public Directory তে আপনার index.php file থাকবে, যেটি আপনার application এর সব ধরনের request এর entry point হিসেবে ব্যবহৃত হবে আর সব ধরনের asset যেমন CSS, JS, fonts, image ইত্যাদি থাকবে।

- **resources**

এই folder এ আপনার ফ্রন্ট এন্ড ফাইল view থাকবে । তা ছাড়া যত রকমের uncompiled যেমন LESS, SASS, JavaScript এবং Language ফাইলগুলোও এখানে থাকবে।

- **routes**

এই folder এ আপনার application এর সব route definitions গুলো থাকবে। যেমনঃ web.php, api.php, channels.php, এবং console.php. আমরা route নিয়ে পরে বিস্তারিত আলোচনা করব।

- **storage**

এই folder এ আপনার application এর সব compiled Blade templates, file based sessions, file caches, এবং framework generated অন্যান্য file গুলো থাকবে। এই folder টি তিনটি folder যথাক্রমে app, framework, এবং logs folder এ বিভক্ত। app folder আপনার application generated ফাইল গুলো থাকবে, framework folder আপনার framework generated ফাইল গুলো থাকবে এবং logs folder আপনার application এর লগ ফাইল গুলো থাকবে।

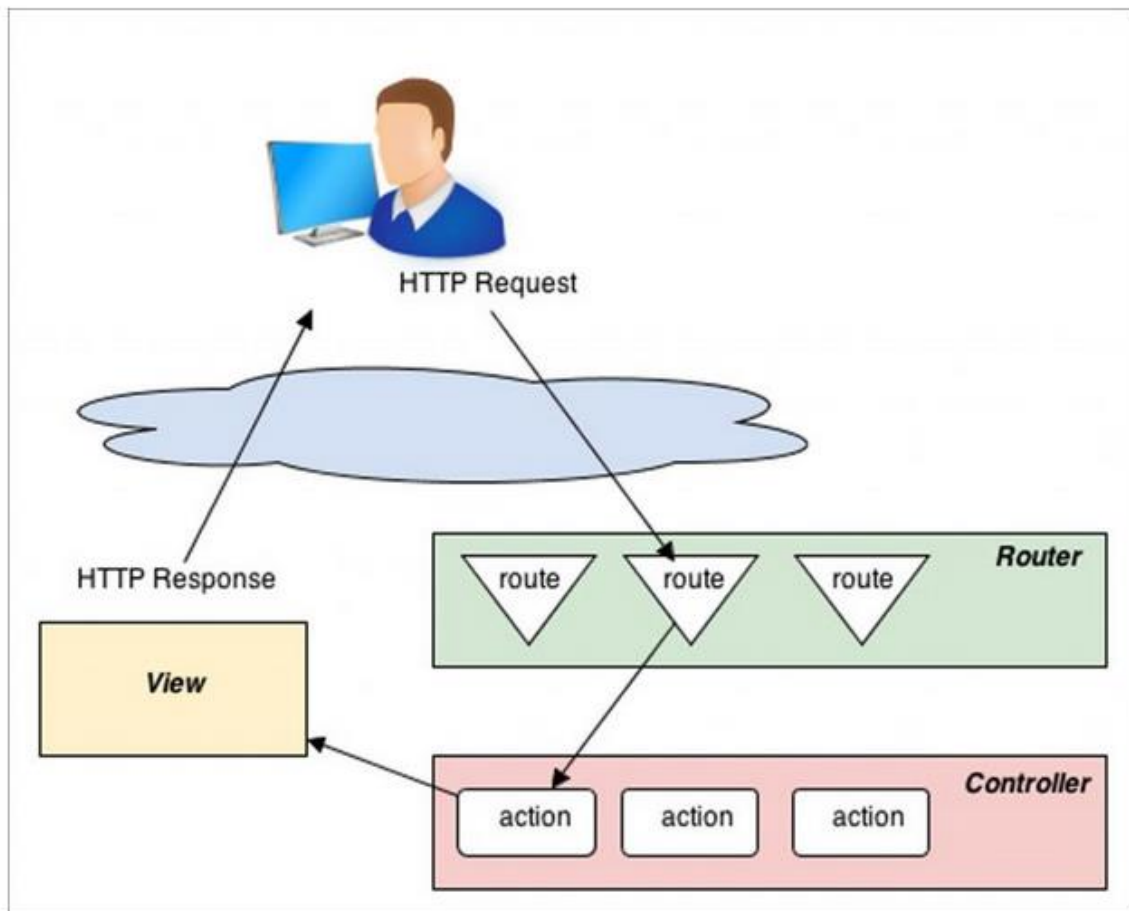
- **tests**

এই folder এ আপনার application এর test ফাইল গুলো থাকে , যেমনঃ PHPUnit ফাইল।

- **vendor**

এই folder এ আপনার সব Composer dependency ফাইলগুলো থাকে।

Laravel Framework এ routes কি?



Laravel Route এর প্রাথমিক ধারণা

Laravel Route লিখার আগে আপনাকে [PHP Anonymous Function, Closure](#) এবং PHP Class এর [Static Method](#) সম্পর্কে ধারণা থাকা দরকার, আমরা ধরে নিলাম আপনি এই গুলো জানেন। তো আসা যাক কিভাবে route দিয়ে URL Request গুলো Manage করতে পারি। ধরুন আমাদের Web Application এ তিনটি পেজ আছে , **সেগুলো যথাক্রমে Home, About এবং Contact.** এখন Web Application ব্যবহারকারী এই তিনটি পেজ এর মধ্যে যাকে request করবে শুধু সংশ্লিষ্ট পেজটি response করবে। অনেকটা নিচের URL এর মত:

for Home page:

http://localhost:8000/

for About page:

http://localhost:8000/about

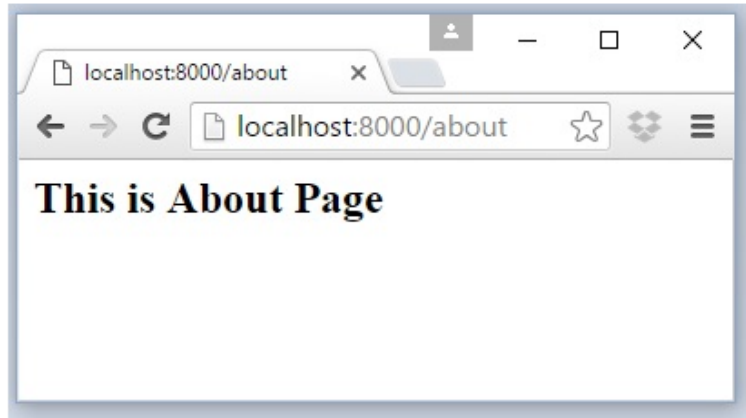
for Contact page:

http://localhost:8000/contact

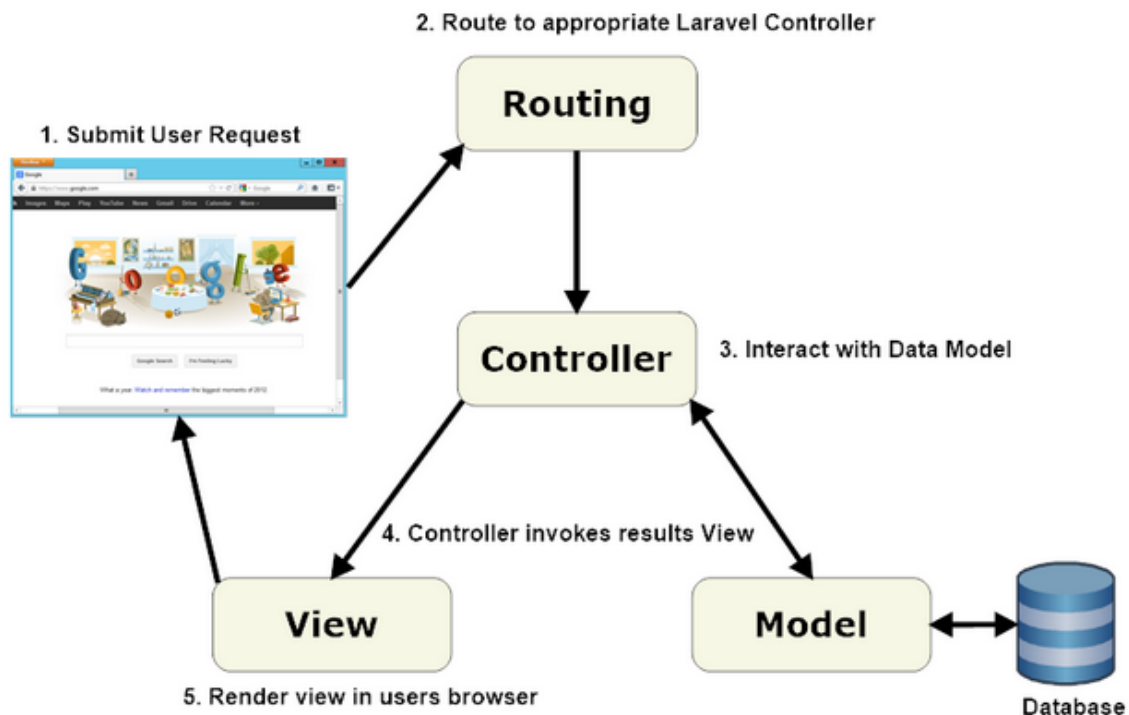
এখন এই তিনটি URL Request এর জন্য আপনি Laravel এর routes/web.php ফাইল এ ঠিক নিচের মত করে লিখতে পারেন।

```
1 Route::get('/', function (){
2     echo "<h2>This is Home Page</h2>";
3 });
4 Route::get('/about', function (){
5     echo "<h2> This is About Page</h2>";
6 });
7 Route::get('/contact', function (){
8     echo "<h2> This is Contact Page</h2>";
9 });
```

এখন আপনি আপনার ব্রাউজার এর Address bar এ `http://localhost:8000` , `http://localhost:8000/about` এবং `http://localhost:8000/contact` লিখে hit করুন, তাহলে আপনার route ঠিক নিম্নের screenshot গুলোর মতো রেজাল্ট দেখাবে:



Larvel Framework এ Controller কি?



1. To create controller named **mainController**:

```
C:\xampp\htdocs>cd your-project-name>php artisan make:controller mainController
```

2. create a function named index in mainController

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class mainController extends Controller
{
    public function index()
    {
        echo "Hello World";
    }
}
```

3. to run controller function from route

```
Route::get('/', 'mainController@index');
```

```
Route::get('/home', 'homeController@home');
```

4. you can send parameter(name,age) by using route to function(home) of a controller

```
Route::get('/home/{name?}/{age?}', 'homeController@home');
```

Laravel route এর মাধ্যমে view folder এর যেকোনো file display করা

1. create a file name **hello.php** in **resources/views** folder

2. in routes/web.php:

```
Route::get('/hello', function (){
    return view("hello");
});
```

3. we can send data:

```
Route::get('/hello', function (){
    $student=[
        "name"=>"Masud Alam"
    ];
});
```

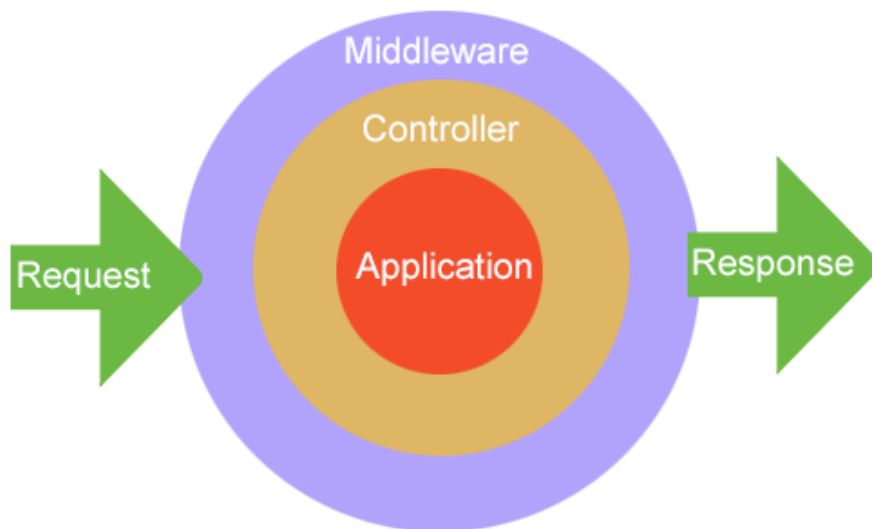


```
    return view("hello",$student);  
});
```

4. to use data in hello.php page:

```
<div class="content">  
    <div class="title m-b-md">  
        <center>Welcome <?=$name;?></center>  
    </div>  
</div>
```

Laravel Framework এ Middleware কি?



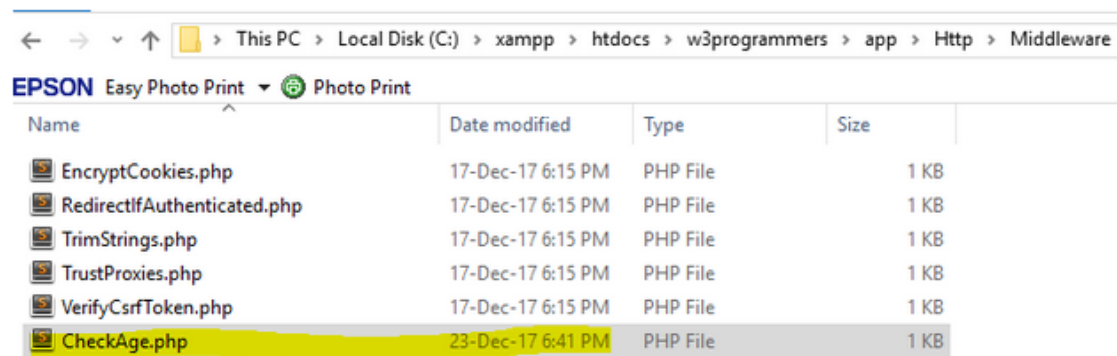
Laravel Middleware

Laravel Framework এ Middleware হচ্ছে আপনার application এ বিভিন্ন সোর্স থেকে আসা HTTP request গুলোকে controller এ পাঠানোর আগে filtering করার একটা দুর্দান্ত Mechanism বা পদ্ধতি। ধরুন Laravel authentication Middleware এর কথাই ধরা যাক, যখন আপনার application এ কোনো user লগইন করতে চায়, সেক্ষেত্রে Laravel authentication Middleware কাজ হচ্ছে যেই user লগইন করতে চায় , তাকে authentication Middleware নিজে Middle Man হিসেবে verify করবে , যদি ইউজার authenticated না হয় , তাহলে Laravel authentication Middleware সেই user কে login screen এ পাঠাবে , আর যদি authenticated হন, তাহলে laravel authentication Middleware ইউজারকে পরবর্তী কাজ করার বা পরবর্তী ধাপে যাওয়ার অনুমতি প্রধান করবে। সহজ ভাবে বলা যায় , আপনার application এ যেকোনো request এর পর এবং response এর পূর্বে সব ধরনের verification এবং authentication করার Mechanism হচ্ছে Laravel Middleware. Laravel এ Middleware ফাইল গুলো app/Http/Middleware এই ডিরেক্টরিতে রাখতে হয়।

1. to create a middleware named CheckAge:

```
php artisan make:middleware CheckAge
```

এখন আপনার app/Http/Middleware ফোল্ডারে checkAge নামে একটা middleware তৈরী হয়ে গেছে ঠিক নিচের মতো :



The screenshot shows a Windows File Explorer window with the address bar displaying the path: This PC > Local Disk (C:) > xampp > htdocs > w3programmers > app > Http > Middleware. The main area shows a list of files with columns for Name, Date modified, Type, and Size. The file CheckAge.php is highlighted in yellow.

Name	Date modified	Type	Size
EncryptCookies.php	17-Dec-17 6:15 PM	PHP File	1 KB
RedirectIfAuthenticated.php	17-Dec-17 6:15 PM	PHP File	1 KB
TrimStrings.php	17-Dec-17 6:15 PM	PHP File	1 KB
TrustProxies.php	17-Dec-17 6:15 PM	PHP File	1 KB
VerifyCsrfToken.php	17-Dec-17 6:15 PM	PHP File	1 KB
CheckAge.php	23-Dec-17 6:41 PM	PHP File	1 KB

এবং আপনার CheckAge Middleware এর কোডটি হবে নিচের মতো :

```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6
7 class CheckAge
8 {
9     /**
10      * Handle an incoming request.
11      *
12      * @param \Illuminate\Http\Request $request
13      * @param \Closure $next
14      * @return mixed
15      */
16     public function handle($request, Closure $next)
17     {
18         return $next($request);
19     }
20 }
```

নতুন CheckAge Middleware রেজিস্ট্রেশন

নতুন checkAge Middleware নিয়ে কাজ করতে হলে আপনাকে app/Http/kernel.php file এ এটিকে রেজিস্টার করে দিতে হবে , ঠিক নিচের মতো :

```
1 protected $routeMiddleware = [
2     'auth' => \Illuminate\Auth\Middleware\Authenticate::class,
3     'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
4     'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
5     'can' => \Illuminate\Auth\Middleware\Authorize::class,
6     'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
7     'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
8     'CheckAge' => \App\Http\Middleware\CheckAge::class,
9 ];
```

Controller Part :

এখন আমরা আমাদের age show করার জন্য একটা Controller তৈরী করব, যেটার আগে middleware চেক করবে http request এর মাধ্যমে আসা age টি valid কিনা তা চেক করবে। চলুন নিচের কমান্ডটির মাধ্যমে একটা showAge নামে Controller তৈরী করা যাক :

```
1 | php artisan make:controller showAge
```

এখন নিচের মতো করে আপনার showAge নামে controller টি তৈরী হবে

```
1 | <?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use Illuminate\Http\Request;
6 |
7 | class showAge extends Controller
8 | {
9 |     //
10 | }
```

এখন showAge Controller এ একটা index method করব যার কাজ হবে http request এর মাধ্যমে আসা age টি দেখাবে। নিচের কোডটি লক্ষ্য করুন :

```
1 | <?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use Illuminate\Http\Request;
6 |
7 | class showAge extends Controller
8 | {
9 |     public function index(Request $request){
10 |         echo "Welcome You're $request->age years Old";
11 |     }
12 | }
13 | }
```

Middleware এ Condition চেক :

এখন আমরা আমাদের তৈরী checkAge Middleware এ আমাদের age Validation করব , ধরুন আমরা আমাদের এপ্লিকেশন এ ৩০ বছরের বেশি বয়সী কেউকে allow করবোনা , এর জন্য আমরা নিচের কোডটি লিখব :

```
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6
7  class CheckAge
8  {
9      /**
10       * Handle an incoming request.
11       *
12       * @param \Illuminate\Http\Request $request
13       * @param \Closure $next
14       * @return mixed
15       */
16     public function handle($request, Closure $next)
17     {
18         if ($request->age>=30) {
19             die("You are $request->age years old, sorry we are unable to");
20         }
21
22         return $next($request);
23     }
24 }
25
```

Route Part :

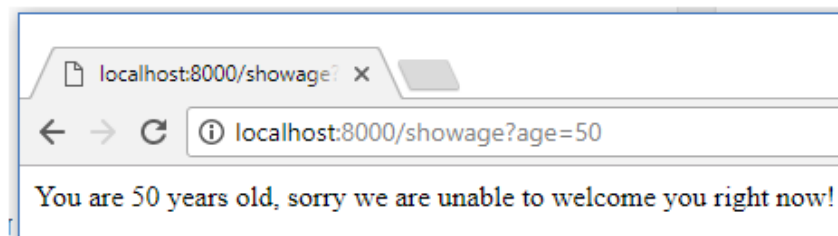
এখন আমরা আমাদের route এ বলে দিবো আমাদের age টি কোন Controller দিয়ে প্রদর্শন করবে , এবং কোন Middleware দিয়ে Validation করবে। চলুন লেখা যাক :

```
1  Route::get("showage",
2      [
3      "uses"=>"showAge@index",
4      "middleware"=>"CheckAge"
5      ]);
```

এখন আমরা ৩০ বছরের বেশি বয়স দিয়ে প্রথমে চেক করব। আর চেক করার জন্য নিচের URL টি আপনার Browser address bar এ দিয়ে hit করুন :

```
1 | http://localhost:8000/showage?age=50
```

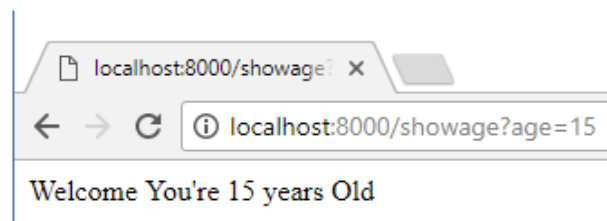
আপনার ব্রাউজার এর ফলাফল নিচের মত আসবে:



আর যদি আপনি age 30 এর কম দেন তাহলে নিচের মতো রেজাল্ট আসবে। এবার নিচের URL টি ব্রাউজার এ দিয়ে হিট করুন :

```
1 | http://localhost:8000/showage?age=15
```

Result



Laravel Middleware Validation Pass

Blade Templating Engine ?

Laravel Framework এ Plain PHP Code থেকে design part যাকে আমরা বলি Presentation Layer (অর্থাৎ HTML, CSS, JS) কে আলাদা করার জন্য Blade Template Engine ব্যবহার করা হয়। আর এটা ব্যবহার করা খুবই সহজ। যদিও Blade আপনাকে আপনার view তে Plain PHP Code ব্যবহার করতে বাধা দেয় না। প্রকৃতপক্ষে, View তে ব্লেডের সব code গুলো Plain PHP কোডে Compiled হয় এবং cache করা হয়। এবং কোডে কোনো সংশোধন করা না হওয়া পর্যন্ত cache অবস্থায় থাকে , যার মানে Blade মূলত আপনার অ্যাপ্লিকেশনে কোনো বাড়তি বোঝা তৈরী করেনা। বরং Plain PHP Code গুলো থেকে HTML,CSS এবং JS কে আলাদা করায় আপনার এপ্লিকেশন এর Security, Performance, code reuse সহ আরো অনেক সুবিধা যোগ করে। Blade Template এর Syntax মূলত [ASP.net Razor syntax](#) সিনট্যাক্স দ্বারা অনুপ্রাণিত। Blade Template এ কোড লেখার জন্য আপনার View ফাইলের এক্সটেনশন হতে হবে .blade.php . Blade Template ফাইল গুলো ও **resources/views** ফোল্ডারে রাখতে হয়।

Blade template তৈরী করার নিয়ম

যেকোনো html Template কে Laravel এর Blade Template এ রূপান্তর করতে হলে প্রথমে আপনাকে আপনার template কে slicing করে header section নিয়ে header.blade.php, sidebar section নিয়ে sidebar.blade.php এবং footer section নিয়ে footer.blade.php ফাইল তৈরী করে নিতে হবে (যদি আপনার html template এ section গুলো থাকে)। আর মূল file কে master.blade.php নামে save করতে হবে। তারপর master.blade.php ফাইল এ Laravel Blade Command @include দিয়ে ফাইল গুলোকে include করে নিতে হবে। তারপর সেই Master Page কে Laravel Blade Command @extends দিয়ে extend করে অন্য view Page গুলো বানাতে হবে। ধরুন আপনার View Page গুলোর নাম যথাক্রমে home,about এবং contact তাহলে Blade এর Format এ এগুলো হবে যথাক্রমে home.blade.php, about.blade.php এবং contact.blade.php.

এবার **Blade Template** এর **File Structure** দেখে নেয়া যাক :

File struture :-

```
-public
--assets
---css
---js
---fonts
---img
- resources
-- views
--- layout
---- master.blade.php
---- header.blade.php
---- footer.blade.php
---- sidebar.blade.php
--- pages
---- home.blade.php
---- about.blade.php
---- contact.blade.php
```

Blade Template Commands

এবার আমরা আমাদের Blade Template এ ব্যবহার করব এমন কিছু Blade Template Commands সম্পর্কে জানা যাক: -

```
@yield :- এটি যেকোনো value প্রদর্শন করতে ব্যবহৃত হয়।
@section :- এটি Template এর যেকোনো section define করতে ব্যবহৃত হয়।
@endsection :- এটি define করা section end করতে ব্যবহৃত হয়।
@include :- এক view file কে অন্য view file এ include করার জন্য ব্যবহৃত হয়।
@extends:- Master Layout কে Child Layout এ extends বা inherit করার জন্য ব্যবহৃত হয়।
{{ asset('your_asset_file_path') }} :- css file, js file এবং যেকোনো image link করার জন্য ব্যবহৃত হয়।
{{URL::to('/your_page_name')}} :-এটা মূলতঃ বিভিন্ন page এর সাথে লিংক করার জন্য ব্যবহৃত হয়
```

master.blade.php

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>@yield('title')</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="Description" lang="en" content="ADD SITE DESCRIPTION">
    <meta name="author" content="ADD AUTHOR INFORMATION">
    <meta name="robots" content="index, follow">

    <!-- icons -->
    <link rel="apple-touch-icon" href="{{ asset('/assets/img/apple-touch-icon.png') }}">
    <link rel="shortcut icon" href="favicon.ico">

    <!-- Bootstrap Core CSS file -->
    <link rel="stylesheet" href="{{ asset('/assets/css/bootstrap.min.css') }}">

    <!-- Override CSS file - add your own CSS rules -->
    <link rel="stylesheet" href="{{ asset('/assets/css/styles.css') }}">

    <!-- Conditional comment containing JS files for IE6 - 8 -->
    <!--[if lt IE 9]>
      <script src="{{ asset('/assets/js/html5.js') }}"></script>
      <script src="{{ asset('/assets/js/respond.min.js') }}"></script>
    <![endif]-->
  </head>
  <body>

    @include('layouts.header')

    <!-- Page Content -->
    <div class="container-fluid">

      <div class="row">
        <div class="col-sm-8">
          <div class="row">
            <div class="col-sm-12">
              <div class="page-header">
                @yield('content')
              </div>
            </div>
          </div>
        </div>
        @include('layouts.sidebar')
      </div>
    <!-- /.row -->
    @include('layouts.footer')
  </div>
  <!-- /.container-fluid -->

  <!-- JQuery scripts -->
  <script src="{{ asset('/assets/js/jquery-1.11.2.min.js') }}"></script>
  <!-- Bootstrap Core scripts -->
  <script src="{{ asset('/assets/js/bootstrap.min.js') }}"></script>

</body>
</html>

```

header.blade.php

```
1 <!-- Navigation -->
2 <nav class="navbar navbar-fixed-top navbar-inverse" role="navigation">
3     <div class="container-fluid">
4
5         <!-- Brand and toggle get grouped for better mobile display -->
6         <div class="navbar-header">
7             <button type="button" class="navbar-toggle collapsed" data-
8                 <span class="sr-only">Toggle navigation</span>
9                 <span class="icon-bar"></span>
10                <span class="icon-bar"></span>
11                <span class="icon-bar"></span>
12            </button>
13            <a class="navbar-brand" href="{{URL::to('/') }}">Home</a>
14        </div>
15        <!-- /.navbar-header -->
16
17        <!-- Collect the nav links, forms, and other content for togglin
18        <div class="collapse navbar-collapse" id="bs-example-navbar-col
19            <ul class="nav navbar-nav">
20                <li><a href="{{URL::to('/about')}} ">About</a></li>
21                <li><a href="{{URL::to('/contact')}} ">Contact</a></li>
22            </ul>
23        </div>
24        <!-- /.navbar-collapse -->
25    </div>
26    <!-- /.container-fluid -->
27 </nav>
28 <!-- /.navbar -->
```

footer.blade.php

এখন resources/views/layouts/ ফোল্ডারে footer.blade.php ফাইলটি তৈরী করব।

```
1 <footer class="margin-tb-3">
2     <div class="row">
3         <div class="col-lg-12">
4             <p>Copyright &copy; w3programmers 2017</p>
5         </div>
6     </div>
7 </footer>
```

sidebar.blade.php

পঞ্চম ধাপ: resources/views/pages ফোল্ডারে Page গুলো বানানো :

সর্বশেষে আমরা আমাদের home.php, about.php, এবং contact.php এই তিনটি file বানাবো।

home.blade.php file

```
1 <p>Posted by <span class="glyphicon glyphicon-user"></span> <a href="#">I
2 @extends('layouts.master')
3 @section('title', 'W3Programmers Home Page')
4 @section('content')
5 <h2>This is my Home page</h2>
6 @stop
```

about.blade.php file

```
1 <p>Posted by <span class="glyphicon glyphicon-user"></span> <a href="#">I
2 @extends('layouts.master')
3 @section('title', 'W3Programmers About Page')
4 @section('content')
5 <h2>This is my About page</h2>
6 @stop
```

contact.blade.php file

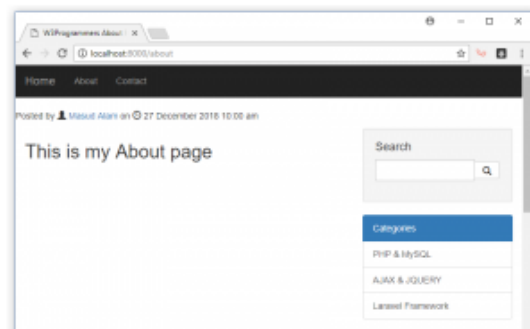
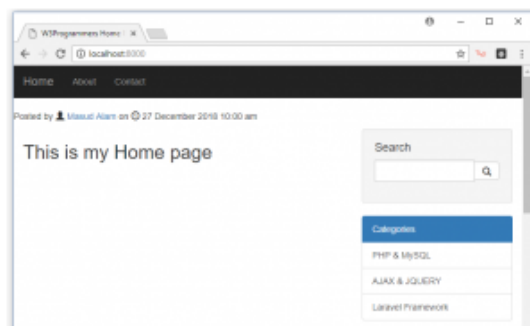
```
1 <p>Posted by <span class="glyphicon glyphicon-user"></span> <a href="#">I
2 @extends('layouts.master')
3 @section('title', 'W3Programmers Contact Page')
4 @section('content')
5 <h2>This is my Contact page</h2>
6 @stop
```

ফাইনাল ধাপ :route এর কাজ

এবার আমরা আমাদের page গুলোকে route এ লিংক করে দিব

```
1 Route::get('/', function () {  
2     return view('pages.home');  
3 });  
4 Route::get('/about', function () {  
5     return view('pages.about');  
6 });  
7 Route::get('/contact', function () {  
8     return view('pages.contact');  
9 });
```

সব কিছু ঠিক থাকলে এখন আপনার Laravel Blade Template রেডি ! আপনি খুব সহজে <http://localhost:8000> , <http://localhost:8000/about> এবং <http://localhost:8000/contact> ব্রাউজ করে নিচের মতো ফলাফল পাবেন।



Laravel Framework এ Migration কি?

প্রযুক্তিগত অগ্রগতির সাথে সাথে , আমরা যারা আই.টি. পেশাদার, ওয়েব ডেভেলপার অথবা প্রোগ্রামার, আমরা অনেকেই 'Database Schema' নামটির সাথে পরিচিত এবং আমরা অনেকেই Database Schema কে create, modify এবং up-to-date রাখতে গিয়ে অনেক সমস্যার সম্মুখীন হই। আপনার Database কে up-to-date রাখা এবং সব জটিলতা সমাধান করার জন্য, Laravel Framework নিয়ে আসছে Migration সুবিধা। Laravel Framework এ Migration ফাইল গুলি /database/migrations ফোল্ডারে থাকে।

Laravel Framework এ, Migrations হচ্ছে আপনার Database Schema গুলোর যেকোনো গঠন পরিবর্তন করার জন্য একটি ফাইল-ভিত্তিক পদ্ধতি। আপনার Database এর Version Control System এর মতো। যা দিয়ে আপনার টিমকে অ্যাপ্লিকেশন গুলোর Database Schema কে সহজেই create, modify এবং share করার সুযোগ দিতে পারেন। এবং current schema কে up-to-date রাখতে পারেন। যেমন, আপনি আপনার Database এ table তৈরি এবং Drop করা, table এ নতুন column যোগ করা, আপডেট করা, মুছে ফেলা, table এ index যোগ করা সহ অন্যান্য কাজের জন্য Laravel Migration ব্যবহার করতে পারেন। পরবর্তীতে যেকোনো ভুলের জন্য অথবা যেকোনো পরিবর্তনের জন্য আপনি খুব সহজে আগের অবস্থায় অথবা পিছনের যেকোনো ভার্সন এ ফিরে আসতে পারেন। কারণ প্রতিটি মাইগ্রেশন একটি টেক্সট ফাইলে সংরক্ষণ করা হয়, আপনি আপনার Project Repository তে রেখে তাদের পরিচালনা করতে পারেন। Laravel Framework এ Migrations সাধারণত Laravel এর schema builder সঙ্গে সংযুক্ত থাকে , যাতে সহজেই আপনার অ্যাপ্লিকেশন এর Database Schema তৈরী করতে পারেন।

ধরুন, আপনি এবং আপনার টিম git Version Control ব্যবহার করেন। হঠাৎ আপনার মনে হলো user টেবিল এ নতুন একটি ফিল্ড লাগবে যার নাম status এবং টাইপ হবে boolean । এখন আপনি Laravel Framework এর migration সুবিধা ব্যবহার করে টেবিলের পরিবর্তন টা করলেন ও git এ পাঠিয়ে দিলেন। এখন আপনার team এর অন্যান্য সদস্যরা চাইলে git merge করে Laravel migration ফাইলটি রান করে Database কে up-to-date করে নিতে পারেন। আগের মতো আর .sql ফাইল গুলো একজন আরেকজনের কাছ থেকে খুঁজে নিতে হবেনা ।

Migration নিয়ে কাজ করার পূর্ব প্রস্তুতি :

Laravel Framework এ Migration নিয়ে কাজ করতে হলে আপনাকে প্রথমে Database এর host এর সাথে connect এবং Database Select করতে হবে। আর এর জন্য আপনাকে আপনার project এর root ফোল্ডারের .env ফাইল এ Database এর host, username, password এবং Database name বলে দিতে হবে। নিচের ছবিটি লক্ষ্য করুন :

```
7 DB_CONNECTION=mysql
8 DB_HOST=127.0.0.1
9 DB_PORT=3306
10 DB_DATABASE=laravel
11 DB_USERNAME=root
12 DB_PASSWORD=' '
13
```

Laravel Database Connection in .env file-

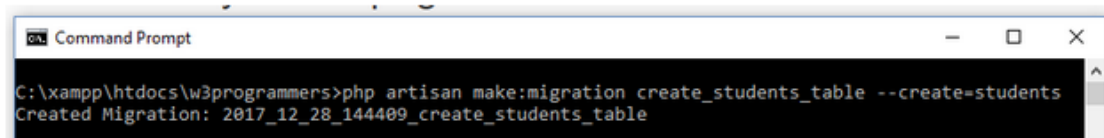
Note: .env file edit করার পর অবশ্যই php artisan সার্ভার টি রিস্টার্ট দিতে হবে। অথবা "php artisan config:clear" এই command টি run করতে হবে।

কিভাবে Laravel Framework এ নতুন Migration File তৈরী করা হয় ?

Laravel Framework এ migration file তৈরী করতে হলে আপনাকে Terminal এ আপনার Project এর root এর ভেতর ঢুকে migration command রান করতে হবে। ধরুন, আমরা students table তৈরী করা জন্য একটা migration file বানাৰ। তাহলে আমাদের CLI তে artisan command টি হবে ঠিক নিচের মত:

```
1 | php artisan make:migration create_students_table --create=students
```

এখন CLI নিচের মতো ফলাফল দেখাবে :



```
Command Prompt
C:\xampp\htdocs\w3programmers>php artisan make:migration create_students_table --create=students
Created Migration: 2017_12_28_144409_create_students_table
```


এখন আপনি database/migrations ফোল্ডারে
2017_12_28_144409_create_students_table.php নামে নতুন তৈরী হওয়া একটা file
দেখতে পাবেন। এবং ফাইলটি খুললে ঠিক নিচের মতো কোডগুলো পাবেন:

```
1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateStudentsTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('students', function (Blueprint $table) {
17             $table->increments('id');
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      *
25      * @return void
26      */
27     public function down()
28     {
29         Schema::dropIfExists('students');
30     }
31 }
```

ব্যাখ্যা : প্রথমত এখানে schema, Blueprint ও Migration নামের তিনটি ক্লাসকে ব্যবহার করা হয়েছে। এবং Migration নামের ক্লাস কে extend করে CreateStudentsTable নামের একটি ক্লাস তৈরী করা হয়েছে। CreateStudentsTable class এ আমরা দুটি Method দেখতে পাই, একটি up অন্যটি down । up মেথড দিয়ে আমরা নতুন table তৈরী, table এ data insert, table এ নতুন field যোগ এবং অন্য যেকোনো পরিবর্তনও করতে পারি। আর down মেথড এ আমরা up মেথডে যে পরিবর্তন টা করেছি সেটা কিভাবে undo করা যায় সেই প্রসেসটা বলে দিতে পারি এবং table drop ও করে দিতে পারি। সাধারনত যেকোনো Table এ একটি id থাকা লাগে এবং table এর যেকোনো রেকর্ড কবে তৈরি ও পরিবর্তন হয়েছে সেইটাও জানতে হয়, তাই Laravel Framework আর্গেই আমাদের জন্য এই বেসিক বিষয় গুলো লিখে দিয়েছে। কিন্তু আমাদের প্রয়োজন না হলে এগুলো মুছে ফেলতে পারি।

Migration ব্যবহার করে Database এ নতুন Table তৈরি করা :

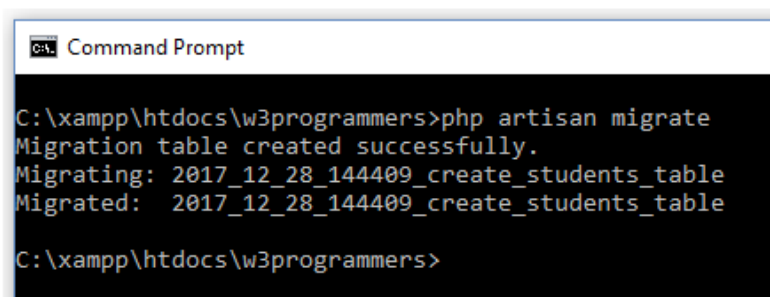
আমরা আমাদের students table টায় আরও কিছু ফিল্ড/কলাম(Column) যোগ করব। আর তার জন্য Laravel blueprint class এর \$table object দিয়ে database table এর জন্য বিভিন্ন data type সহ \$table object এর বিভিন্ন Method call করব। লাইন নম্বর ১৬ থেকে ২১ দেখুন :

```
1  <?php
2  use Illuminate\Support\Facades\Schema;
3  use Illuminate\Database\Schema\Blueprint;
4  use Illuminate\Database\Migrations\Migration;
5
6  class CreateStudentsTable extends Migration
7  {
8      /**
9       * Run the migrations.
10      *
11      * @return void
12      */
13     public function up()
14     {
15         Schema::create('students', function (Blueprint $table) {
16             $table->increments('id');
17             $table->string('name', 20);
18             $table->string('email',50);
19             $table->string('mobile',20);
20             $table->tinyInteger('status')->default(1);
21             $table->timestamps();
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      *
28      * @return void
29      */
30     public function down()
31     {
32         Schema::dropIfExists('students');
33     }
34 }
```

এবার আমাদের বানানো এবং আগে থাকা(যদি থাকে) Migration গুলোকে দিয়ে আমরা আমাদের Database এ table গুলি বানিয়ে ফেলি। চলুন Terminal এ নিচের কমান্ডটি রান করি :

```
1 | php artisan migrate
```

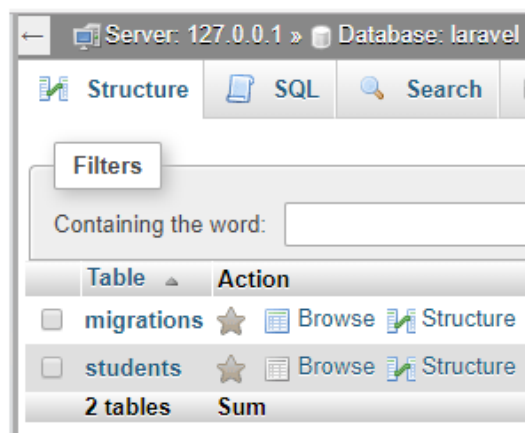
আপনার CLI নিচের মত ফলাফল দেখাবে :



```
C:\xampp\htdocs\w3programmers>php artisan migrate
Migration table created successfully.
Migrating: 2017_12_28_144409_create_students_table
Migrated: 2017_12_28_144409_create_students_table
C:\xampp\htdocs\w3programmers>
```

Laravel Migration Complete

উপরের screenshot দেখে বুঝা যাচ্ছে যে আমাদের Migration হয়ে গেছে। এখন আপনি আপনার Database টি দেখুন। এখানে আমাদের students table ছাড়াও আরও একটি টেবিল আছে যার নাম migrations. এটি লারাভেল নিজে ব্যবহার করে। নিচে আমাদের phpmyadmin screenshot result দেওয়া হলো:

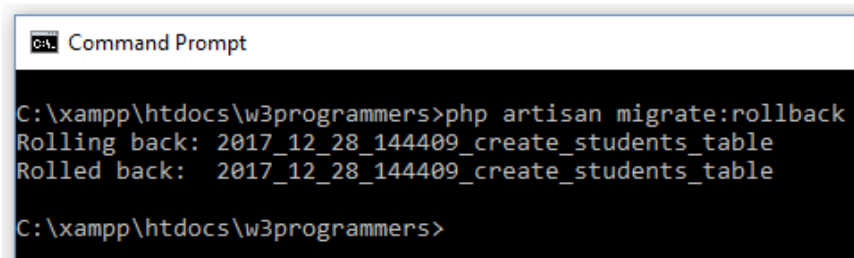


মাইগ্রেশনটি rollback বা undo করা:

কখনো আমরা আমাদের database এর আগের অবস্থানে যেতে হতে পারে , ধরুন আপনি একটা Migration run করার পর আপনার মনে হলোঃ আপনার যেকোনো একটি column নামের বানান ভুল হয়েছে বা আরও একটি ফিল্ড add করতে হবে তখন আবার নতুন মাইগ্রেশন না লিখে rollback আপনি rollback করতে পারেন, তারপর আপনি আপনার মাইগ্রেশনটাকে প্রয়োজন মতো লিখে আবার মাইগ্রেন্ট কমান্ড রান করতে পারবেন। আর এর জন্য আপনাকে নিচের command টি রান করতে হবে :

```
1 | php artisan migrate:rollback
```

এখন আপনাকে ঠিক নিচের মত রেজাল্ট দেখাবে :



```
cmd. Command Prompt
C:\xampp\htdocs\w3programmers>php artisan migrate:rollback
Rolling back: 2017_12_28_144409_create_students_table
Rolled back: 2017_12_28_144409_create_students_table
C:\xampp\htdocs\w3programmers>
```

এখন আপনি আপনার table টা আর database এ দেখতে পাবেন না, অর্থাৎ , Larvel আপনার সর্বশেষ করা students table রিমুভ করে দিবে। এবং Laravel তার নিজস্ব table migrations টি রেখে দিবে।

Note:আপনার কোনও মাইগ্রেশনে যদি Rename অথবা Drop থাকে তাহলে উপরের command টি error দিবে। এই জন্য, doctrine/dbal নামক একটি প্যাকেজ আপনার প্রোজেক্ট এ যোগ করতে হবে। Terminal এ শুধু নিচের কমান্ডটি রান করুনঃ

```
1 | composer require doctrine/dbal
```

এটা আমাদের সব প্যাকেজ গুলোকে update, প্রয়োজনে add করে নিবে।

Migration ব্যবহার করে সব টেবিল গুলো মুছে ফেলা :

Laravel Framework এ Migration ব্যবহার করে সব table মুছে ফেলতে চাইলে আপনি নিচের কমান্ডটি রান করাই করতে পারেন :

```
1 | php artisan migrate:reset
```

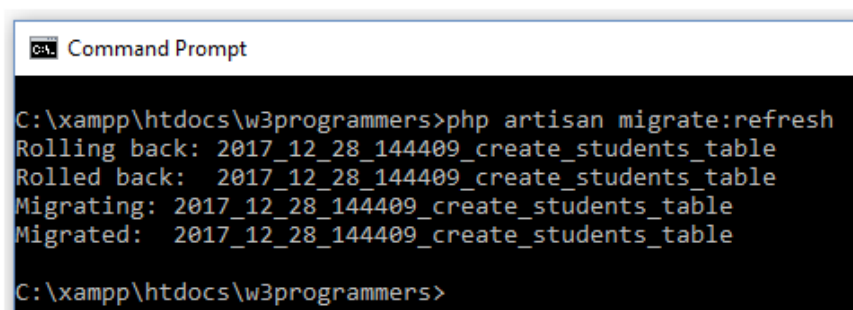
এখন পুরা ডাটাবেজ ক্লিয়ার হয়ে যাবে এবং আবার যদি মাইগ্রেট কমান্ড রান করাই তাহলে একে বারে ফ্রেশ একটি ডাটাবেজ পাবো।

একবারে সব টেবিল মুছে ফেলে নতুন করে তৈরি

একবারে সব টেবিল মুছে ফেলে নতুন করে তৈরি করতে চাইলে আপনি নিচের কমান্ডটি রান করাই করতে পারেন :

```
1 | php artisan migrate:refresh
```

এখন আপনাকে ঠিক নিচের মত রেজাল্ট দেখাবে :



```
C:\xampp\htdocs\w3programmers>php artisan migrate:refresh
Rolling back: 2017_12_28_144409_create_students_table
Rolled back: 2017_12_28_144409_create_students_table
Migrating: 2017_12_28_144409_create_students_table
Migrated: 2017_12_28_144409_create_students_table
C:\xampp\htdocs\w3programmers>
```

laravel migration refresh

Laravel Query Builder দিয়ে Database Table এর Result গুলো retrieve করা :

Controller Class

```

<?php
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;
class getqueryController extends Controller
{
    public function index(){
        $students = DB::table('students')->get();
        return view('students', ['students' => $students]);
    }
}

```

students.php

```

<!DOCTYPE html>
<html>
<head>
    <title>Students View</title>
</head>
<body>
<?php
foreach ($students as $student) {
    echo $student->name, "<br>";
}
?>
</body>
</html>

```

Controller Class

Database এর একটি নির্দিষ্ট table থেকে একটি single row অথবা single column retrieve করা

আপনি Database এর table থেকে একটি single row অথবা single column retrieve করার প্রয়োজন হলে, আপনি first Method ব্যবহার করতে পারেন। আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
<?php
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;

class getqueryController extends Controller
{
    public function index(){
        $user = DB::table('students')->where('name', 'Masud Alam')->first();
        echo $user->name;
    }
}
```

আপনি সম্পূর্ণ row এর পরিবর্তে সরাসরি একটি column এর value print করতে চাইলে value() method টি ব্যবহার করতে পারেন। এর জন্যে আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
use App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;

class getqueryController extends Controller
{
    public function index(){
        $email = DB::table('students')->where('name', 'Masud Alam')->value('email');
        echo $email;
    }
}
```

public function show(\$id)

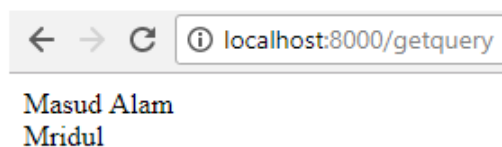
```
{
    $car = Car::find($id);
    return view('cars.show', array('car' => $car));
}
```

Table এর একটি নির্দিষ্ট column এর সব row একসাথে retrieve করা :

Table এর একটি নির্দিষ্ট column এর সব row একসাথে retrieve করতে চাইলে আপনি DB facade এর pluck() Method ব্যবহার করতে পারেন। নিচের উদাহরণটি দেখুন :

```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9          $names = DB::table('students')->pluck('name');
10         foreach ($names as $name) {
11             echo $name, "<br>";
12         }
13     }
14 }
```

Result



তবে আপনি চাইলে table এর একটি column কে key এবং অন্যটিকে value হিসেবে রিটার্ন করতে পারেন। নিচের উদাহরণটি দেখুন :

```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9          $names = DB::table('students')->pluck('name','id');
10         foreach ($names as $id=>$name) {
11             echo "ID: ", $id;
12             echo " Name: ", $name,"<br>";
13         }
14     }
15 }
```

Result



ID: 1 Name: Masud Alam

ID: 2 Name: Mridul

Chunking Results

Database এ হাজার হাজার record নিয়ে কাজ করতে গেলে আপনাকে অনেক সময় সব ডাটা একসাথে retrieve করতে বিভিন্ন সমস্যায় পড়তে হয়, এর জন্য আপনি DB facade এর chunk method ব্যবহার করতে পারেন। নিচের উদাহরণটি দেখুন :

```
<?php
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;

class getqueryController extends Controller
{
    public function index(){
        DB::table('students')->orderBy('id')->chunk(2, function ($students) {
            echo "<pre>";
            print_r($students);
            echo "</pre>";
        });
    }
}
```

Result

```
Illuminate\Support\Collection Object
(
    [items:protected] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 1
                    [name] => Masud Alam
                    [email] => masud@mail.com
                )

            [1] => stdClass Object
                (
                    [id] => 2
                    [name] => Mridul
                    [email] => mridul@mail.com
                )
        )
)

Illuminate\Support\Collection Object
(
    [items:protected] => Array
        (
            [0] => stdClass Object
                (
                    [id] => 3
                    [name] => Habib
                    [email] => habib@mail.com
                )

            [1] => stdClass Object
                (
                    [id] => 4
                    [name] => Rashad
                    [email] => rashad@mail.com
                )
        )
)
```

Laravel Framework chunk

Aggregates

Laravel Query Builder count, max, min, avg, এবং sum নামে কিছু aggregate Method এর সুবিধা দেয়, নিচে এগুলোর কিছু উদাহরণ দেওয়া হলো :

```
1  $users = DB::table('users')->count();  
2  $price = DB::table('orders')->max('price');  
3  $price = DB::table('orders')  
4              ->where('finalized', 1)  
5              ->avg('price');
```

একটি নির্দিষ্ট column এর সবগুলো row কে show করা

Laravel Framework এ একটি নির্দিষ্ট column এর সবগুলো row কে show করার জন্য আপনি **select()** method টি ব্যবহার করতে পারেন। আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
<?php
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;

class getqueryController extends Controller
{
    public function index(){
        $students = DB::table('students')->select('name', 'email as
        student_email')->get();
        $table="<table border='1' width='300'";
        $table.="<tr><th>Name</th><th>Email</th></tr>";
        foreach($students as $student){

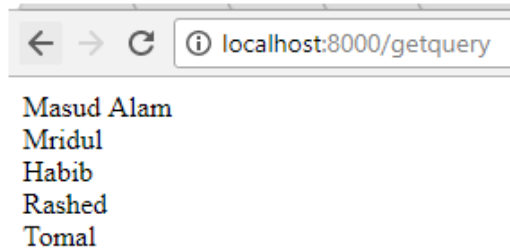
            $table.="<tr><td>$student->name</td>
            <td>$student->student_email</td></tr>";

        }
        $table.="</table>";
        echo $table;
    }
}
```

Laravel Framework এ একটি নির্দিষ্ট column এর duplicate value গুলো বাদ দিয়ে unique value যুক্ত row গুলো কে select করার জন্য আপনি distinct() method টি ব্যবহার করতে পারেন। আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9
10         $students = DB::table('students')->distinct()->get(["name"]);
11         foreach($students as $student){
12             echo $student->name."<br>";
13         }
14     }
15 }
```

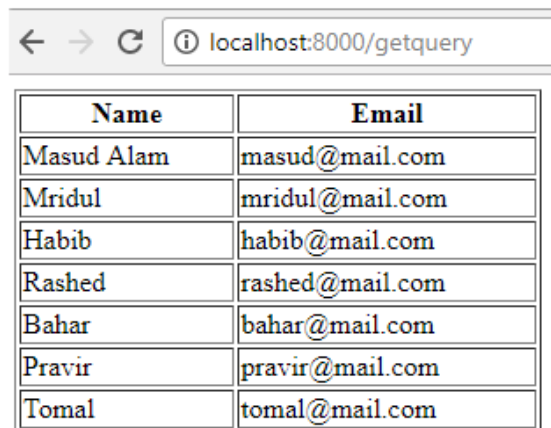
এখন আপনি যদি আপনার ব্রাউজার এ <http://localhost:8000/getquery> হিট করেন তাহলে নিচের মত ফলাফল দেখাবে



existing select clause এর সাথে আরো নতুন column add করার জন্য আপনি addSelect() Method টি ব্যবহার করতে পারেন। আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9
10         $query = DB::table('students')->select('name');
11         $students = $query->addSelect('email')->get();
12
13         $table="<table border='1' width='300'";
14         $table.="<tr><th>Name</th><th>Email</th></tr>";
15         foreach($students as $student){
16
17             $table.="<tr><td>$student->name</td><td>$student->email</td></tr>";
18
19         }
20         $table.="</table>";
21         echo $table;
22     }
23 }
```

এখন আপনি যদি আপনার ব্রাউজার এ <http://localhost:8000/getquery> হিট করেন তাহলে নিচের মত ফলাফল দেখাবে



The screenshot shows a web browser window with the address bar displaying 'localhost:8000/getquery'. Below the address bar, there is a table with two columns: 'Name' and 'Email'. The table contains seven rows of student data.

Name	Email
Masud Alam	masud@mail.com
Mridul	mridul@mail.com
Habib	habib@mail.com
Rashed	rashed@mail.com
Bahar	bahar@mail.com
Pravir	pravir@mail.com
Tomal	tomal@mail.com

Inner Join Clause

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Support\Facades\DB;
```

```
use Illuminate\Http\Request;
```

```
class getqueryController extends Controller
```

```
{
```

```
    public function index(){
```

```
        $students = DB::table('students')
```

```
            ->join('contacts', 'students.id', '=', 'contacts.student_id')
```

```
            ->join('course_choice', 'students.id', '=', 'course_choice.student_id')
```

```
            ->join('courses', 'students.id', '=', 'course_choice.student_id')
```

```
        -
```

```
        >select('students.id','students.name','courses.course_name','contacts.phone','contacts.email')
```

```
            ->get();
```

```
        echo "<pre>";
```

```
        print_r($students);
```

```
        echo "</pre>";
```

```
    }
```

```
}
```

Left Join Clause

আপনি যদি inner join এর পরিবর্তে left join করতে চান , তাহলে আপনাকে join method এর পরিবর্তে leftJoin Method টি ব্যবহার করতে হবে। এর জন্য leftJoin Method এর প্রথম argument এ আপনাকে Database এর table নাম দিতে হবে, আর বাকি argument গুলোতে যেইসব column এর ভিত্তিতে leftJoin করব তার নাম দিতে হবে আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;

class getqueryController extends Controller
{
    public function index(){
        $students = DB::table('students')
            ->leftJoin('contacts', 'students.id', '=', 'contacts.student_id')
            ->select('students.id','students.name','contacts.phone','contacts.email')
            ->get();

        echo "<pre>";
        print_r($students);
        echo "</pre>";
    }
}
```

Cross Join Clause

আপনি যদি cross join করতে চান , তাহলে আপনাকে crossJoin Method টি ব্যবহার করতে হবে। আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Support\Facades\DB;
4 use Illuminate\Http\Request;
5
6 class getqueryController extends Controller
7 {
8     public function index(){
9
10         $students = DB::table('students')
11             ->crossJoin('courses')
12             ->get();
13
14         echo "<pre>";
15         print_r($students);
16         echo "</pre>";
17     }
18 }
```


Advanced Join Clauses

আপনি চাইলে Laravel Query Builder দিয়ে আরো advance join query রান করতে পারেন, আর এর জন্য আপনার join Method এর দ্বিতীয় Argument হিসেবে একটা closure পাঠাতে হবে। তখন closure টি একটি joinClause অবজেক্ট রিসিভ করবে, যেটি পরবর্তী constraints কে allow করবে। আরো ভালো ভাবে বুঝার জন্য আপনার getqueryController class এ নিচের কোডটি লিখুন :

```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9
10         $students=DB::table('students')
11         ->join('contacts', function ($join) {
12             $join->on('students.id', '=', 'contacts.student_id')
13             ->where('contacts.student_id', '>', 3);
14         })
15         ->get();
16
17         echo "<pre>";
18         print_r($students);
19         echo "</pre>";
20     }
21 }
```

Unions

Laravel Framework এ Query Builder দিয়ে আপনি দুটি query এর মধ্যে খুব সহজে Union করতে পারেন। আরো ভালো ভাবে বুঝার জন্য আপনার getqueryController class এ নিচের কোডটি লিখুন :

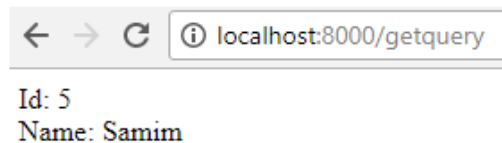
```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9
10         $first = DB::table('contacts')
11             ->whereNull('phone');
12
13         $students = DB::table('contacts')
14             ->whereNull('email')
15             ->union($first)
16             ->get();
17
18         echo "<pre>";
19         print_r($students);
20         echo "</pre>";
21     }
22 }
```

Simple Where Clauses

একটা উদাহরণ দিয়ে ব্যাপারটা আরো ভালোভাবে বুঝা যাক, ধরুন আমরা আমাদের students table থেকে শুধু ৫ নম্বর id এর স্টুডেন্ট কে দেখাব, তাহলে আমাদেরকে Laravel এর where method কে getqueryController class এ নিচের মত লিখতে হবে।

```
1 <?php
2 namespace App\Http\Controllers;
3 use Illuminate\Support\Facades\DB;
4 use Illuminate\Http\Request;
5
6 class getqueryController extends Controller
7 {
8     public function index(){
9
10         $students = DB::table('students')->where('id', '=', 5)->get();
11         foreach($students as $student){
12             echo "Id: ",$student->id,"<br>","Name: ",$student->name;
13         }
14     }
15 }
```

এখন আপনি যদি আপনার ব্রাউজার এ <http://localhost:8000/getquery> হিট করেন তাহলে নিচের মত ফলাফল দেখাবে



Id: 5
Name: Samim

তবে Laravel where clause এ অন্যান্য where clause এর মত আপনি <= >= <> != like এইসব ব্যবহার করতে পারেন। নিচের উদাহরণগুলো দেখুন :

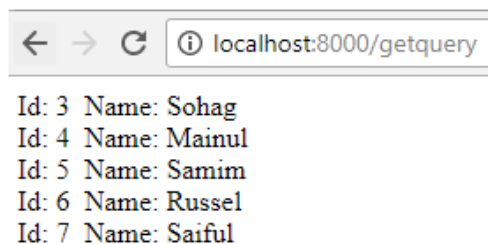
```
1 $students = DB::table('students')->where('id', '=', 5)->get();
2 $students = DB::table('students')->where('id', '!=', 5)->get();
3 $students = DB::table('students')->where('id', '<=', 5)->get();
4 $students = DB::table('students')->where('id', '>=', 5)->get();
5 $students = DB::table('students')->where('id', '<>', 5)->get();
6 $students = DB::table('students')->where('id', 'like', '%S')->get();
7 $students = DB::table('students')->where([
8     ['status', '=', '1'],
9     ['subscribed', '<>', '1'],
10 ])->get();
```

Or Statements

Laravel Framework এ অনেকগুলো শর্তের মধ্যে আপনি যেকোনো একটি পাওয়ার জন্য `orWhere` ব্যবহার করতে পারেন। আপনার `getQueryController` class এ নিচের কোডটি লিখুন :

```
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Support\Facades\DB;
4  use Illuminate\Http\Request;
5
6  class getqueryController extends Controller
7  {
8      public function index(){
9
10         $students = DB::table('students')
11             ->where('id', '>', 3)
12             ->orWhere('name', 'Sohag')
13             ->get();
14         foreach($students as $student){
15             echo "Id: ",$student->id,"&nbsp;&nbsp;&nbsp;","Name: ",$student->name,"<br>";
16         }
17     }
18 }
```

এখন আপনি যদি আপনার ব্রাউজার এ `http://localhost:8000/getquery` হিট করেন তাহলে নিচের মত ফলাফল দেখাবে



```
Id: 3 Name: Sohag
Id: 4 Name: Mainul
Id: 5 Name: Samim
Id: 6 Name: Russel
Id: 7 Name: Saiful
```

সব গুলো where clauses method গুলো নিচে table আকারে দেওয়া হল :

Method নাম	Method এর ব্যবহার	উদাহরণ
whereBetween	একটি নির্দিষ্ট column এর value টি দুটি value এর range এর মধ্যভর্তি যেকোন value কিনা, তা যাচাই করে।	<pre>1 \$students = DB::table('students') 2 ->whereBetween('id', [1, 5]) 3 ->get();</pre>
whereNotBetween	একটি নির্দিষ্ট column এর value টি দুটি value এর range এর মধ্যভর্তি যেকোন value নয়, তা যাচাই করে।	<pre>1 \$students = DB::table('students') 2 ->whereNotBetween('id', [1, 5]) 3 ->get();</pre>
whereIn	একটি নির্দিষ্ট column এর value গুলো আপনার দেওয়া যেইসব array এর value গুলোর সাথে মিলবে , শুধু সেগুলো নিয়ে কাজ করবে।	<pre>1 \$students = DB::table('students') 2 ->whereIn('id', [1,2,3]) 3 ->get();</pre>

whereNotIn	একটি নির্দিষ্ট column এর value গুলো আপনার দেওয়া যেইসব array এর value গুলোর সাথে মিলবে না , শুধু সেগুলো নিয়ে কাজ করবে।	<pre> 1 \$students = DB::table('students') 2 ->whereNotIn('id', [1,2,3]) 3 ->get(); </pre>
whereNull	একটি নির্দিষ্ট column এর মান null কিনা তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereNull('updated_at') 3 ->get(); </pre>
whereNotNull	একটি নির্দিষ্ট column এর মান null নয় তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereNotNull('updated_at') 3 ->get(); </pre>
whereDate	একটি নির্দিষ্ট column এর মান একটি নির্দিষ্ট date কিনা তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereDate('created_at', '2017-12-31') 3 ->get(); </pre>
whereMonth	একটি নির্দিষ্ট column এর মান একটি নির্দিষ্ট Month কিনা তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereMonth('created_at', '12') 3 ->get(); </pre>
whereDay	একটি নির্দিষ্ট column এর মান একটি নির্দিষ্ট day কিনা তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereDay('created_at', '31') 3 ->get(); </pre>

whereYear	একটি নির্দিষ্ট column এর মান একটি নির্দিষ্ট year কিনা তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereYear('created_at', '2017') 3 ->get(); </pre>
whereTime	একটি নির্দিষ্ট column এর মান একটি নির্দিষ্ট time কিনা তা চেক করে।	<pre> 1 \$students = DB::table('students') 2 ->whereTime('created_at', '12:30') 3 ->get(); </pre>
whereColumn	দুটি column এর মান এর মধ্যে compare করার জন্য ব্যবহার করা হয়।	<pre> 1 \$students = DB::table('students') 2 ->whereColumn('first_name', 'last_name') 3 ->get(); </pre> <hr/> <pre> 1 \$users = DB::table('users') 2 ->whereColumn(3 'updated_at', '>', 'created_at' 4)->get(); </pre> <hr/> <pre> 1 \$students = DB::table('students') 2 ->whereColumn([3 ['first_name', '=', 'last_name'], 4 ['updated_at', '>', 'created_at'] 5])->get(); </pre>

Parameter Grouping

কখনো কখনো আপনাকে advance query লেখার জন্য অনেক গুলো where এর Parameter গুলোকে Grouping করতে হতে পারে, Laravel query builder এই কাজটা দারুন ভাবে হ্যান্ডেল করেছে , Parameter Grouping বুঝার জন্য নিচের query টি লক্ষ্য করুন :

```
1 DB::table('students')
2     ->where('name', '=', 'Habib')
3     ->orWhere(function ($query) {
4         $query->where('id', '>', 5)
5             ->where('title', '<>', 'Mr');
6     })
7     ->get();
```

উপরের query টি নিচের মত SQL তৈরী করবে :

```
1 | select * from students where name = 'Habib' or (id > 5 and title <> 'Mr')
< |----->
```

Where Exists Clauses

whereExists Method দিয়ে আপনি where exists clause তৈরী করতে পারেন। whereExists Method টি মূলত closure এর argument রিসিভ করার মাধ্যমে কাজ করে থাকে। আর closure এর argument টি query builder class এর instance receive করে , যা query define করার কাজে ব্যবহৃত হয়। Where Exists Clauses বুঝার জন্য নিচের query টি লক্ষ্য করুন :

```
1 DB::table('users')
2     ->whereExists(function ($query) {
3         $query->select(DB::raw(1))
4             ->from('orders')
5             ->whereRaw('orders.user_id = users.id');
6     })
7     ->get();
```

উপরের query টি নিচের মত একটি query generate করবে।

```
1 | select * from users
2 | where exists (
3 |     select 1 from orders where orders.user_id = users.id
4 | )
```


JSON Where Clauses

Laravel Framework এ MySQL 5.7 এবং PostgreSQL Database এর json column type কলাম থেকে আপনি Data query করতে পারেন। আর এর জন্য আপনাকে "->" Operator টি ব্যবহার করতে হবে। JSON Where Clauses বুঝার জন্য নিচের query টি লক্ষ্য করুন :

```
1 $users = DB::table('users')
2     ->where('options->language', 'en')
3     ->get();
4
5 $users = DB::table('users')
6     ->where('preferences->dining->meal', 'salad')
7     ->get();
```

Conditional Clauses

একটি নির্দিষ্ট condition true হওয়ার ভিত্তিতে where clause রান করতে হয়, আর এর জন্য আপনাকে when condition clause ব্যবহার করতে হবে। when condition clause ব্যবহার বুঝার জন্য নিচের উদাহরণটি দেখুন :

```
1 $role = $request->input('role');
2
3 $users = DB::table('users')
4     ->when($role, function ($query) use ($role) {
5         return $query->where('role_id', $role);
6     })
7     ->get();
```

ব্যাখ্যা :উপরের closure টি শুধু তখনই execute হবে , যখন first parameter এর condition true হবে। আর first parameter false হলে closure execute হবেনা।

Ordering

```
$students= DB::table('students')
    ->orderBy('name', 'desc')
    ->get();
```

groupBy / having

```
$students = DB::table('students')
    ->groupBy('student_id')
    ->having('student_id', '>', 5)
    ->get();
```

skip / take

```
$students = DB::table('students')->skip(10)->take(5)->get();
```

Limit

```
$students = DB::table('students')
    ->offset(10)
    ->limit(5)
    ->get();
```

Insert, Update and Delete with Laravel

Inserts:

```
$data=DB::table('students')->insert(
    ['name' => 'Masud Alam', 'email' => 'masud@w3prorammers.com']);

echo $data?"Insert Success":"Data Insert Fail";
```

Get Inserted ID:

```
public function index(){
    $id = DB::table('students')->insertGetId(
        ['name' => 'Farhan', 'email' => "farhan@donald.com"]
    );

    echo "Your Last Inserted Id: $id";
```

Updates

```
$data=DB::table('students')
    ->where('id', 1)
    ->update(['name' => 'Kabir Hossain']);

echo $data?"Data Update Success":"Data Update Fail";
```

Updating JSON Columns

```
DB::table('users')
    ->where('id', 1)
    ->update(['options->enabled' => true]);
```

Increment & Decrement

```
$data=DB::table('students')->increment('visits', 2);

echo $data?"Data Update Success":"Data Update Fail";
```

Deletes

```
public function index(){
    $result=DB::table('courses')->delete();
    $result2=DB::table('students')->where('id', '=', 4)->delete();
    echo $result?"All Data Delete Success<br>":" All Data Delete Fail<br>";
    echo $result2?"Data Delete Success":" Data Delete Fail";
}
```

আপনি যদি পুরো টেবিলের মুছে ফেলতে চান, অর্থাৎ , যা সব সারি মুছে ফেলবে এবং auto increment id reset করবে , তাহলে আপনি Query Builder এর truncate Method ব্যবহার করতে পারেন, আরো ভালোভাবে বুঝার জন্য নিচের কোডটি দেখুন :

```
1 | DB::table('users')->truncate();
```

মডেল তৈরি:

C:\xampp\htdocs>your-project-name>**php artisan make:model Student**

মডেল গুলো সাধারণ ভাবে app ডিরেক্টরির রুটেই থাকে, মডেলটি খুললে আমরা এরকম পাবোঃ

```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Student extends Model
8  {
9      //
10 }
```

Select

```
public function viewAlldata($data)
{
    DB::table('users')->select('*')->get();
}
```

Insert

```
public function Insertdata($data)
{
    DB::table('tablename')->insert($data);
}

public function store(Request $request)
{
    //

    $student = new Student;

    $student->name = $request->name;
    $student->email=$request->email;
    $student->mobile=$request->mobile;
    $student->status=$request->status;
    $student->created_at=now();
    $student->updated_at=now();

    echo $student->save()?"Insert Success":"Insert Fail";
}
```

```

public function create()
{
    //

    $student = new Student;

    $student->name = "Masud Alam";
    $student->email = "masud.eden@gmail.com";
    $student->mobile = "01788229988";
    $student->status = 1;
    $student->created_at = now();
    $student->updated_at = now();

    echo $student->save() ? "Insert Success" : "Insert Fail";

}

$users = [
    [
        'name' => str_random(10),
        'email' => str_random(10).'@gmail.com',
        'password' => bcrypt('secret'),
    ],
    [
        'name' => str_random(10),
        'email' => str_random(10).'@gmail.com',
        'password' => bcrypt('secret'),
    ],
    [
        'name' => str_random(10),
        'email' => str_random(10).'@gmail.com',
        'password' => bcrypt('secret'),
    ]
];

DB::table('users')->insert($users);

```

Updates

```

Student::where('active', 1)
    ->where('subject', 'English')
    ->update(['delayed' => 1]);

```

```

public function update(Request $request, Student $student)
{
    //
    $student=Student::find(1);
    $student->name="Rashed Alam";
    $student->email="rashed@mail.com";
    $student->mobile="01799330022";
    echo $student->save()?"Update Success":"Update Fail";
}

```

Delete

```

public function destroy($id)
{
    $deletedRows = Student::where('id', $id)->delete();
}

```

////////Model////////

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\User;

class UserController extends Controller

```

{
    public function __construct()
    {
        $this->middleware('auth');
    }
    public function index()
    {
        //
        $users = User::latest()->paginate(5);
        return view('users.all', compact('users'));
    }
}

```

```

public function create()
{
    //
}
public function store(Request $request)
{
    //
}
public function show($id)
{
    //
}
public function edit($id)
{
    //
}
public function update(Request $request, $id)
{
    //
}
public function destroy($id)
{
    //
}
}

```

Form in Laravel :

[resources/views/form.php](#)

```

<html>
<body>

<?php
echo Form::open(array('url' => 'foo/bar'));

    echo Form::text('username','Username');
    echo '<br/>';

    echo Form::text('email', 'example@gmail.com');
    echo '<br/>';

    echo Form::password('password');
    echo '<br/>';

```

```

        echo Form::checkbox('name', 'value');
        echo '<br/>';

        echo Form::radio('name', 'value');
        echo '<br/>';

        echo Form::file('image');
        echo '<br/>';

        echo Form::select('size', array('L' => 'Large', 'S' => 'Small'));
        echo '<br/>';

        echo Form::submit('Click Me!');
        echo Form::close();
    ?>

</body>
</html>

```

app/Http/routes.php

```

Route::get('/form',function(){
    return view('form');
});

```

Visit the following URL to see the form:

<http://localhost:8000/form>

Laravel File Upload with Validation

resources>views>uploadfile.blade.php

```
@extends('layouts.app')
```

```
@section('content')
```

```
<div class="container">
```

```
<div class="row justify-content-center">
```



```
<div class="card">
```

```
<div class="card-header">Upload File Example</div>
```

```
<div class="card-body">
```

```
@if ($message = Session::get('success'))
```

```
<div class="alert alert-success alert-block">
```

```
<button type="button" class="close" data-dismiss="alert">×</button>
```

```
<strong>{{ $message }}</strong>
```

```
</div>
```

```
@endif
```

```
@if (count($errors) > 0)
```

```
<div class="alert alert-danger">
```

```
<strong>Whoops!</strong> There were some problems with your input.<br><br>
```

```
<ul>
```

```
@foreach ($errors->all() as $error)
```

```
<li>{{ $error }}</li>
```

```
@endforeach
```

```
</ul>
```

```
</div>
```

```
@endif
```

```

        <form action="/uploadfile" method="post" enctype="multipart/form-data">
            @csrf
            <div class="form-group">
                <input type="file" class="form-control-file" name="fileToUpload"
id="exampleInputFile" aria-describedby="fileHelp">
                <small id="fileHelp" class="form-text text-muted">Please upload a valid image
file. Size of image should not be more than 2MB.</small>
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
</div>
</div>
</div>
</div>
@endsection

```

Link Menu

```

<li class="nav-item"><a class="nav-link" href="{{ url('/uploadfile') }}">Upload File</a></li>

```

Route

```

Route::get('uploadfile','UserController@uploadfile');

```

Controller

```

public function uploadFile(){
    return view('uploadfile');
}

```

File Upload function in Controller

```
public function uploadFilePost(Request $request){  
    $request->validate([  
        'fileToUpload' => 'required|file|max:1024',  
    ]);  
    $request->fileToUpload->store('logos');  
    return back()->with('success','You have successfully upload image.');
```

Route

```
Route::post('uploadfile','UserController@uploadFilePost');
```

Laravel Multiple File Upload with JQuery



[create.blade.php //file upload form](#)

```

<html lang="en">
<head>
<title>Laravel Multiple File Upload Example</title>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.js"></script>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
</head>
<body>
<div class="container">
    @if (count($errors) > 0)
    <div class="alert alert-danger">
        <strong>Whoops!</strong> There were some problems with your input.<br><br>
        <ul>
            @foreach ($errors->all() as $error)
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
    @endif

    @if(session('success'))
    <div class="alert alert-success">
        {{ session('success') }}
    </div>
    @endif

    <h3 class="jumbotron">Laravel Multiple File Upload</h3>
    <form method="post" action="{{url('file')}}" enctype="multipart/form-data">
        {{csrf_field()}}

        <div class="input-group control-group increment" >
            <input type="file" name="filename[]" class="form-control">
            <div class="input-group-btn">
                <button class="btn btn-success" type="button"><i class="glyphicon glyphicon-
plus"></i>Add</button>
            </div>
        </div>

        <div class="clone hide">
            <div class="control-group input-group" style="margin-top:10px">
                <input type="file" name="filename[]" class="form-control">
                <div class="input-group-btn">
                    <button class="btn btn-danger" type="button"><i class="glyphicon glyphicon-
remove"></i> Remove</button>
                </div>
            </div>
        </div>
    </form>

```

```
</div>
```

```
<button type="submit" class="btn btn-primary" style="margin-top:10px">Submit</button>
```

```
</form>
```

```
</div>
```

```
<script type="text/javascript">
```

```
$(document).ready(function() {
```

```
    $(".btn-success").click(function(){  
        var html = $(".clone").html();  
        $(".increment").after(html);  
    });
```

```
    $("body").on("click", ".btn-danger",function(){  
        $(this).parents(".control-group").remove();  
    });
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```

Create a **Model** and **Controller** by below command

```
php artisan make:model File
```

```
php artisan make:controller FileController
```

এটি নিম্নের দুটি ফাইল তৈরি করবে।

```
1 | File.php
2 | FileController.php
```

এখন routes>> web.php ফাইল এ নিম্নোক্ত route গুলি যোগ করুন।

```
1 | // web.php
2 | Route::get('file','FileController@create');
3 | Route::post('file','FileController@store');
```

FileController এর create function এ নিম্নলিখিত কোডটি লিখুন

```
1 | /**
2 |     * Show the form for creating a new resource.
3 |     *
4 |     * @return \Illuminate\Http\Response
5 |     */
6 | public function create()
7 | {
8 |     //
9 |     return view('create');
10 | }
```

এখন যদি আপনি আপনার ব্রাউজার এর URL তে `http://localhost:8000/file` লিখে হিট করেন , তাহলে নিম্নের ছবির মতো একটি multiple file upload form দেখতে পাবেন।

Laravel Multiple File Upload

Choose File

No file chosen

+Add

Choose File

No file chosen

✕ Remove

Submit

ধাপ-৩: Backend Validation যুক্ত করা

যেহেতু আমরা একাধিক ফাইল Insert করার ব্যবস্থা রাখছি, তাই আমরা আমাদের প্রজেক্টের মধ্যে array validation প্রয়োজন। FileController.php ফাইলের store() function এর মধ্যে আমাদের ইনপুট ফাইল Validation করতে নিম্নলিখিত কোড যুক্ত করুন।

```
1  /**
2      * Store a newly created resource in storage.
3      *
4      * @param \Illuminate\Http\Request $request
5      * @return \Illuminate\Http\Response
6      */
7  public function store(Request $request)
8  {
9      $this->validate($request, [
10         'filename' => 'required',
11         'filename.*' => 'mimes:doc,pdf,docx,zip'
12     ]);
13 }
```

এটি required field এর পাশাপাশি file type ও চেক করবে। যদি input file এ file গুলোর মধ্যে doc,pdf, docx অথবা zip file না হয় তাহলে লারাভেল আপনাকে front-end এ একটি error দেখাবে। আর ফর্ম এ error দেখানোর জন্য আপনার create.blade.php file এ নিচের কোড টি লিখুন:

create.blade.php

```
@if (count($errors) > 0)
<div class="alert alert-danger">
    <strong>Whoops!</strong> There were some problems with your input.<br><br>
    <ul>
        @foreach ($errors->all() as $error)
            <li>{{ $error }}</li>
        @endforeach
    </ul>
</div>
@endif
```

ধাপ ৪: ডাটাবেস এ Multiple File সংরক্ষণ

File Validation Check করার পর, আমারদেরকে ডাটাবেসের মধ্যে ফাইলের নাম সংরক্ষণ করতে হবে। তাই ডাটাবেসে বিভিন্ন ফাইলের নাম সন্নিবেশ করার জন্য আমাদের চূড়ান্ত কোডটি নিম্ন রূপ হবে।

```

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\File;

class FileController extends Controller
{
    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
        return view('create');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param \Illuminate\Http\Request $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $this->validate($request, [

            'filename' => 'required',
            'filename.*' => 'mimes:doc,pdf,docx,zip'

        ]);

        if($request->hasfile('filename'))
        {
            foreach($request->file('filename') as $file)
            {
                $name=$file->getClientOriginalName();
                $file->move(public_path().'/files/', $name);
                $data[] = $name;
            }
        }

        $file= new File();
        $file->filename=json_encode($data);

        $file->save();

        return back()->with('success', 'Your files has been successfully added');
    }
}

```



```
////////////////////////////////////
```

To create ImageUploadController

php artisan make:controller ImageUploadController

```
use App\ImageUpload;

public function fileStore(Request $request)
{
    $image = $request->file('file');
    $imageName = $image->getClientOriginalName();
    $image->move(public_path('images'),$imageName);

    $imageUpload = new ImageUpload();
    $imageUpload->filename = $imageName;
    $imageUpload->save();
    return response()->json(['success'=>$imageName]);
}
```

Set Route

routes >> web.php

```
Route::get('image/upload','ImageUploadController@fileCreate');
Route::post('image/upload/store','ImageUploadController@fileStore');
Route::post('image/delete','ImageUploadController@fileDestroy');
```

এবার ImageUploadController.php ফাইলটিতে যেতে হবে এবং fileCreate() ফাংশনে নিম্নের কোড যোগ করতে হবে:

```
1 //
2 public function fileCreate()
3 {
4     return view('imageupload');
5 }
```

imageupload.blade.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Laravel Multiple Images Upload Using Dropzone</title>
```

```

<meta name="_token" content="{{csrf_token()}}" />
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/dropzone/5.4.0/min/dropzone.min.css">
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/dropzone/5.4.0/dropzone.js"></script>
</head>
<body>
<div class="container">

    <h3 class="jumbotron">Drag and Drop Files here to upload</h3>
    <form method="post" action="{{url('image/upload/store')}}" enctype="multipart/form-data"
        class="dropzone" id="dropzone">

        @csrf
    </form>
    <script type="text/javascript">
        Dropzone.options.dropzone =
        {
            maxFileSize: 12,
            renameFile: function(file)
            {
                var dt = new Date();
                var time = dt.getTime();
                return time+file.name;
            },
            acceptedFiles: ".jpeg,.jpg,.png,.gif",
            addRemoveLinks: true,
            timeout: 5000,
            success: function(file, response)
            {
                console.log(response);
            },
            error: function(file, response)
            {
                return false;
            }
        };
    </script>
</body>
</html>

```