# Module-7 Live Class One - Mastering Laravel -Structure Artisan, Routing, and Best Practices

Time -  05 Nov , 9 pm

Time Stamp

Start : 20:00

[[Laravel]] : 24 :00

 [[Laravel Documentation]] : 27:20

[[Laravel Ecosystem ]]: 29:30

[[Laravel Create Project Command]]: 32::16

[[.env ]]: 37:24

[[Artisan]] : 38:39

[[Php Artisan Serve]] : 40 : 35

[[MVC]] : 41:45

[[Route]] : 43:40

[[Controller Making ]]: 44 :17

[[Quick File Opening / Finding ]]: 45 : 48

[[Controller Code Writing ]]: 46 :25

[[Route]] : 47 : 52

[[PHP Devsense]] : 48:30 ( Route Running)

[[Taking Parameter From User In Route]] : 53:12

[[Optional Parameter]] : 56:34

[[Working In api.php File]] : 57:35

[[Http Post Request]] : 1:00:00

[[Postman Too]]l : 1:03:00

[[Postman Post Request Error]] : 1:05:00

[[Posting Using Form]] : 1:08:00

[[Posting Using Postman]] : 1:10:00

[[How To Send Data In View ]]: 1:10:38

[[A Small Project ]] ; 1:19:40

[[location service url ]] :  https://wttr.in/dhaka?format=j1

[[Using CSS Inside Laravel ]]: 1:28:00

[[Passing The Query String ]] : 1:38:00

Question Answer Part : 1:40

[[Using Multiple Parameter In Route ]] : 1:41:00

[[Dependency Injection]] : 1:44:00 && 1:53:00

[[Laravel Performance Bottleneck ]] : 1:46:36

[[Project Scope]] : 1:51:00

# Module -7 Conceptual Class - One - MVC Architecture , Request Lifecycle , Introduction to Laravel Project's Structure , Mastering in Laravel Routes

Time : 06 Nov

Time Stamp

Start : 19:20

[[MVC Architecture]] : 21:36

[[Introduction To Laravel Project Structure]] : 27:35

[[Request Lifecycle ]]: 1:17:36 (Laravel Documentation :  Architectural Concept )

[[Mastering Laravel Routes]] : 1:31:57 (Please Check Route Concept With Documentation)

[[Introduction To Laravel View ]]:  goest side by side with 1:31:57

[[Introduction To Laravel Controller]] : 2 : 21 : 00

[[Introduction To Laravel Model]]  :  2:32:57 ( Will be covered in later class.)

[[Introduction To Artisan Commands]] :  2:33:33

[[Postman Simulation Of API Routes ]] :  Kept For Next Class

Question & Answer : 2:36: 40

[[Command For Creating Blade File & Controller Inside A Folder]] : 2:37:27

[[Common Error Of Not Including Class]]  : 2:54:59

# Module-7 Live Class Two -Comprehensive Terminology - HTTP Clients , URIs , Encryption, JWT - CSR or SSR

Time - 7th Nov

Time Stamp

Start : 15:00

[[Request Response Model ]]: 27:05 ( Resource Directory Is Client Side , And Rest All Are Server Side In Laravel )

[[Controller Concept]] : 47:10

[[Middleware Concept]] : 49:25

[[JSON In Details]] :  53:00

[[HTTP Client]] : 1:03:00

[[Postman Http Client]] :  1 : 09 : 27

[[When Use GET()  and When POST()]]  : 1:20:00

[[Cookies Concept]] : 1:22:20

[[Encryption and Decryption]] : 1:31:04

[[JSON Web Token Concept]] : 1:32:44

[[Server Side Rendering ]] : 1:47:00

[[Client Side Rendering]] : 1:49:10

[[Pros & Cons Of CSR & SSR]]: 1:51:20

[[Optimized Solution ]] : 2:06:30

Question & Answer : 2:07:20

[[Single Page Application ]] : 2 : 11 : 00

[[Basic Strong With PHP ]] : 2 : 16 : 48 ( Learn Curl , OOP  With PHP)

[[Basic Strong With PHP ]] : 2 : 16 : 48 ( Learn Curl , OOP  With PHP)

[[Laravel Converts Into Associative Array]] : 2 : 20 :12

[[Use  jwt.io website to understand encryption]] : 2 : 24 : 12

[[Laravel Vs React Or Vue Js]] : 2 : 30 : 00

# Module -7 Conceptual Class - Two - Summary

Time : 08 Nov

Time Stamp

Start : 15:00

[[Making VerifyCSRFToken Class Inactive ]] :  43 :32

[[Using Web Route To Post Data ]] : 43:32

[[Register One External Route File ]] :  47:50  ( app service provider and route service provider is important )

[[JWT Token]] :  59 : 20

Quest & Answer : 1 : 14 : 00

# Module-8 Live Class One - Handling Request & Data In Web Development

Time : 12  Nov

Time Stamp

Start : 12 : 33

[[return response used ]] : new today

[[Blank Project , Controller, Routing ]] : 13 : 00

[[Route Forcing Or Using Constraint  Like whereAlpha , whereNumber & whereAlphaNumeric]] : 19:15

[[Multiple Parameter In Route]] : 22 : 58

[[QueryString]]: 27 : 41

[[Default Value From Input]] : 30:05

[[Post Request Access]] : 33 : 09

[[CSRF Token]] : 36:40

[[Image & File Posting ]] :  40 :50 ( there are some continued discussion on new view file and passing data which was relevant)

[[enctype = "multipart/form-data"]] : 54 : 09

[[Difference Between asset Helper And public_path Helper]] : 57 : 42

[[Post Request Using API ]] : 59:40

[[JSON Accepting Using Postman ]]: 59 : 40  ( same context as before )

Recap  : 1:28:23

Question & Answer :  1 : 12 : 00
[[GitLense Extension]] : 1 : 18 : 48

# Module-8 Conceptual Class - One

Date : 13 Nov , 10 PM

Start : 32 : 35

Image Upload : 35 : 40 ( See Rabbil Vai  Video)

Pre Recorded Must Watch : 37:50

Public Vs Resource Folder : 40 : 37

Image Naming : 42 : 36

Request Object  : 45 : 52 ( Rabbil Vai Video)

Time Function : 48 : 10

Postman Issues : 50 : 14

Request Response Model : 53 : 00  ( See Module 7 Conceptual Class )

Package Removing Methods : 53 : 45

Hosting  Root Folder : 55 : 37

Index File Priority : 57 : 45

Laravel Project Clone : 1 : 00 : 00

Request  / Response  Header File Dealings : 1 : 07 : 45

Network Tab On Screen Header Cookies Showing : 1 : 12 : 00

Data Passing From One Page To Another : 1 : 19 : 49  ( suppose ticket booking app)

Laravel Storage Folder : 1 : 21 : 12

Huge Sized Image Auto Cad File  Storage : 1 : 29 : 09

Rest API : 1 : 37 : 55

VPS / Cloud : 1 : 41 :55

Discussion : 1 : 43 : 00

# Module-8 Live Class Two - Exploring Various Response Formats in Web Development

Time : 14  Nov

Time Stamp

Start : 12:00

[[Request Response Model ]]: 13 : 35

[[String Int Null Boolean Response]] : 22 : 17

[[Web Socket Protocol]] : 26  : 30 ( Just They Mentioned The Name )

[[Array & Associative Array Response]] : 27 : 00 :

[[Json Response]] :  32 : 00 && 44 : 00

[[Response With Data Message Code]] : 34 : 00

[[Status Codes]] : 38 : 05

[[Response Redirecting]] : 45 : 50

[[Binary File Response]] : 48 : 15  ( not showing actual path while movie downloading )

[[File Download Response]] :  57 : 07

[[Response With Cookies]] : 59 : 00

[[Response With  Header Properties]] : 1 :  20 :  45

[[Response With Blade View]] : 1 : 22 : 48

[[Rest API Related Things]]  :  1 : 25 : 45

Question & Answer : 1 : 35 : 25

[[Is Return Type Declaring Optional ]]: 1 : 35 : 37  ( Clean Code Practice Is Declaring Return Type )

[[Weak Warning]] : 1 : 36 : 25

[[Why Should We Use Header]] : 1 : 41 : 33

[[Associative Array Becomes Json Array]] : 1 : 44 : 00

[[Json Response Error Codes ]] :  1 : 46 : 33

[[Some Discussion On Liveware , React & Inertia]] : 1 : 54 : 00

[[Json Response Best Practice Using Ajax Or Axios ]] :  1 : 57 : 30  ( exception error handle always think that it will be from user side : User Is The Biggest QA )

[[You Can Use Try Catch Error Block But Never Allow Any Error To Catch ]] : 2 : 06 : 53 ( Inside Application Alway Use Codes That Will Work Always In Every Execution , Do Not Wait For Error Catch  )

[[Model Naming Convention Of Laravel ]] : 2 : 09  : 09

# Module-8 Conceptual Class-Two

Date : 16 Nov , 10 PM

Start : 35 :00

[[Associative Array Converts Into What]] : 49 : 18 ( Laravel Associative Array Means Json Object )

[[Nested Looping]] : 51 : 52 ( You need to avoid nested looping as much as you can )

[[Route Declaring ]]: 54 : 40 ( @ using before function It was of laravel 7 )

[[Object Accessor Or Arrow Operator ]] : 57 : 15 ( See PHP Method Chaining )

[[Name Function In Route]] : 1 : 13 : 50

[[{{url ( ' ')}}]] : 1 : 27 : 10 ( use route helper for naming and api , url helper if you put any asset type of files outside of public folder )

[[Validating And Storing Both Inside Controller Using Request ]]: 1 : 30 : 30 ( Single Responsibility )

[[Stroe_Post_ _Request To Separate Validating And Storing]] : 1 : 32 : 39

[[Storing Data Into Several Tables]] : 1 : 35 : 06 ( db transaction is must )

[[Single Responsibility Or Code Separation ]]: 1 : 39 : 44

[[KernelPhp -> Verify CSRF Token Off]] : 1 : 41 : 22

[[Life Time Access]] : 1 : 42 : 00

[[Request Header ]]: 1 : 50 : 05

[[Backend Developer Don't Need Visualization ( Website ) ]] : 2 : 01 : 49

[[What To Learn]] : 2 : 04 : 00

[[NUXT]] : 2 : 10 : 15

[[Inertia JS]] :  2 : 12 : 00

[[Form Field : 2 : 16  : 45]]  ( $request->validate is a good option )

[[Custom Route]] :  2 : 27 : 00  ( Not recommended unless you have a huge project)

[[One Controller Should Not Hold More Than 7 Methods ]]  2 : 30 : 56 ( Resource Controller Make 7  By Default )

# Module -9 Live Class One - Mastering Middleware and Controllers in Laravel

Time : 19 Nov  9 PM

Time Stamp

Start : 15:00

[[Middleware ]]: 17 : 00

[[Rate Limiting / Throttling ]]:  20 : 20

[[Middleware Continued ]] : 25 : 25

[[Middleware : Group Route  ]]: 34 : 00

[[Middleware Usage ]] : 35 : 45

[[Request & Response Process]] :  37 : 40

[[Country Blocker : Middleware]] : 49 : 10

[[Getting IP  ( $request ->ip()) ]] : 51 : 15

[[Country Blocker Continued ]] : 52 : 00

(Resource -  ipinfo.io , cloudflare.warp, cloud flare automatically selects location  )

[[$request -> header('cf-ipcountry') ]] : 56 :00

( Hasin Haydar Vai  : Hydra  Package )

[[Laravel Sanctum]] : 1: 06 : 00

[[Laravel Pulse]] : 1 : 07 : 00

[[Laravel Pail]] : 1 : 07 : 25  ( Logs File )

[[Middleware Mostly Used For Authentication]] :  1 : 11 : 34

Question & Answer : 1 : 12 : 15

[[Closure $next Explained]] : 1 : 13 : 36

[[Why Throttle Is Used]] : 1 : 15 : 51  ( Based On One Individual User )

[[Single Sign On / Auth O]]  : 1 : 19 : 10

[[Passport  & Sanctum For Role Management]] : 1 : 23 : 10

[[Nested Group Middleware]] : 1 : 26 : 00

[[How To Do Multi-language]] :  1 : 27 : 15 ( Custom Is The Best Way Though)

[[Same Middleware For Both Web & API ]] :  1 : 28 : 00

[[Custom Throttle / Rate Limited]] :  1 : 31 : 14  ( Sometimes We See Slowness While Giving Wrong Username Or Password )

[[Hosting In Cloudflare ]] : 1 : 32 : 00 ( DNS will be cloud-flare , local geolocation  )

# Module -9 Conceptual Class One - Middleware, Basic Controller , Single Action Controller , Resource Controller

Date :  20 Nov , 10 PM

Start  :   20  : 30

[[Middleware]] : 28 : 45

[[Kernel.php]] : 38 :00

[[Middleware Command ]] : 43 : 15

[[Controller Method]]. : 44 : 00

[[Middleware Registering]] :  48 : 00

[[How Middleware Activates Before Controller ]] : 48 : 40

[[Group Middleware ]] : 51 : 30

[[Using __construct Inside Controller To Call Middleware ]] : 56 : 34

[[->except( 'middlewareName')]] : 58 : 00

[[->only('middlewareName')]] : 59 : 30

[[Sending Middleware In Array Where Many ]] : 1 : 01 : 33

[[Group Middleware & Nested Inside ]]  1 : 26 : 00

[[Request Verification $request->path()]]  : 1 : 02 : 02

[[Route :: redirect() , Request Redirect Using Middleware ]] : 1 : 04 : 30

[[Header Data Check ]] : 1 : 06 : 40

[[Request Header Data Adding  ]] : 1 : 11 : 00

[[$request->header->add(['access-token'=>'1234']) Inside Middleware ]] : 1 : 12 : 00

[[Controller]] : 1 : 21 : 15

[[Resource Controller ]] : 1 : 28 : 20

[[Single Action Controller ]] : 1 : 33 : 40

[[--invokable ]] : 1 : 33 : 40

[[Using Middle For Whole Application ]] : 1 : 40 : 20

[[All Web Route Middleware Registering ]] : 1 : 41 : 00

[[All API Route Middleware Registering]] : 1 : 42 : 10

Question And Answer : 1 : 43 : 35

[[__construct Inside Controller ]] : 1 : 44 : 00 ( auto run while controller is called, substitute of group middleware )

[[Only & Except ]] : 1 : 47 : 30

[[Resource Route Does Not Take Any Parameters ]] : 1 : 49 : 30

[[Do Not Try For Custom Route Inside Kernel.PHP As A Beginner ]] : 1 : 55 : 18

[[Name Changed Issue For A Student ]]: 1 : 59 : 11  : Model & Controller Same Name Issue

[[Route Via Middleware Or Controller ]] :  2 : 02 : 30

[[Prefix In Middleware ]] :  2 : 04 : 00

princenoman.hashnode.dev /different-ways-of-routing-in-laravel

[[Omitting Middleware Problems ]] : 2 : 07 : 45

[[How To Keep Attention -Syntax Practice By Writing ]] : 2 : 10 : 55

[[Note Taking In Digital Format ]]:  2 : 10 : 05

[[->withoutMiddleware()]] : 2 : 17 : 25 ( This is helpful inside one group middleware if anyone uses the prefix and also don't want to use the middleware )

[[Only & Except]] : 2 : 18 : 35  (inside group middleware you can disable using these directives)

[[Closure $next ]] : 2 : 30 : 20 (Closure takes the request to controller)

[[Headers vs Header ]] 2 : 33 : 00  (While We Add Header Then Headers Keyword , But When Fetching Values From Header Then Header Keyword)

[[Merge Middleware]] : 2 : 34 : 15

[[Throttle]] : 2 : 38 : 00

[[Component Based Layout Vs Extended Layout ]] : 2 : 39 : 29

Reusable  Common Ones -> Component  There are Two Types Normal Component & Class Based Component

We will use both Hybrid Models Of Components .

# Module-9 Live Class Two-Leveraging Laravel Features For Web Development

Date : 21 Nov , 9 PM

Start  :  14 :30

[[View Part]] : 18 : 00  ( Client Side , Inside Resource Folder This Stays)

[[Blade Template Engine]] : 18  : 20

[[Only Rest API Dev Or API Dev ]]: 19 : 45  ( Someone Who Works Only With Route , Controller , Middleware And Model )

[[Names Of Other Template Engine]] : 20 : 33  ( Asp.net - Blazor , Express- Mouses, React - Jsx )

[[Build In Attributes In Blade]] : 21 : 25

[[Front End Files]] : 22 : 25

[[Using Raw PHP ]]: 23 : 00

[[Return View In Controller ]]: 25 : 00

[[Using Raw PHP Continued ]]: 27 : 25 ( All Business Logic Should Be Inside Controller)

[[Why Not Logic Codes Inside View Files]] : 27 : 25 ( Presentation Layer's Job Is Up To Response)

[[@ - All Directives]] : 30 : 00

[[Blade Writing Rules]] : 31 : 10 (html and css rendered and then creates DOM so JS only works on that , that is the reason why JS codes will be in the lower part )

[[{{}} ]]: 33 : 30 (you should not write codes massively inside curly braces)

[[Comments]] : 35 : 15

[[Adding Asset]] : 36 : 10 (you can work fine without asset helper but while going live this will not recognize the root directory as been changed from localhost )

[[Hyperlink]] : 40 : 20

[[Components Architecture]] : 42 : 00

[[Master Layout]] : 44 : 49 (Usually Two Type Before Login and After Login)

[[Hero Section]] : 47 : 20

[[Adding Subview Or Section]] : 47 : 45

[[Props Drilling / Sending Data From Parent To Child]] : 51 : 20 ( Parent Associative Array Sending and Child Variable While Getting Controller Is Grandfather , Master Layout Father , Homepage Is Son ---- )

[[Conditional Rendering ]] : 1 : 01 : 20

[[@unless]] : 1 : 11 : 35 ( Opposite Of If )

[[@for and @foreach ]]: 1 : 11 : 55

(If view renders something that is server side rendering , if you go and click view page source you might see all the html there)

Master Layout Concept : 1 : 25 : 25  (Entire Application title , head and body tag will be just in one place only )

[[Using ChatGPT To Know The Usages Of Directives ]] : 1 : 42 : 15

Question & Answer : 1 : 45 : 30

[[Keeping Design Part Separate ]]:  1 : 48 : 30

[[Using JS]] : 1 : 49 : 25

[[Blade View Structure Example]] : 1 : 50 : 00  (Should Be Loosely Coupled )

[[Yield Can Be Used Many Times]] : 1 : 54 : 30

[[Parent Child]] : 1 : 56 : 16

[[JS Below]] : 2 : 02 : 56

[[Controller Preference]] : 2 : 05 : 51  (do not use many many folders for controllers , only when you have 25/30 controllers then you can arrange them )

[[Third Party Packages Are Good And Proven]] : 2 : 07 : 00

# Module-9 Conceptual Class Two - Laravel Blade Template Engine, Laravel Master Layout Architecture

Date :  22 Nov , 10 PM

Start :  12 : 27

[[Best Practice]] : 16 : 00

[[Throttle For Whom]] : 17 : 40

[[Make A Sample Dashboard Template For Future]] : 27 : 00  (search google free template admin dashboard)

[[Mastering Dashboard]] : 47 : 00  ( we wont load common items again and again)

Views will hold folders like Layouts , Pages and Components
Components can be common items like : navigation , header , footer , sidebar etc
Master layout can be named as mater.blade or app.blade

[[Nice Method Of Asset File Linking ]]: 1 : 11 : 20

Master Layout Continued : 1 : 14 : 12

[[Graphical Explanation Of Master Layout ]] :  1 : 24 : 44

[[Clean Code For Blade File]] : 1 : 55 : 00

All business logic should be inside the controller cause you might need to convert your blade into react , view and and as api for scaling . so do not unnecessarily generate logic inside the blade. Scaling is a must do trait for the modern world . The blade is a presentation layer . Here data will be shown by getting data from controllers not generated here unless very small amounts .

[[Asset File Linking Nicely Continued ]]:  1 : 58 : 00

[[Dashboard Explained ]]: 2 : 06 : 00

# Module-10 Live One -Database Migration , Essentials - Conventions , Structure , Types & Relationships

Date : 26 Nov , 9 PM

Start : 14 : 40

[[Laravel Migration Can Do php my admin Job ]] : 16 : 10

[[Database Design]] : 16 : 30  (application got slower due to database)

[[.env And Migration ]] : 17 : 55

[[Making Alias Of Php My Admin Command]] : 21 : 10

```
  alias art = "php artisan"
 art make: migration create_profiles_table
 table names should be plural by convention
```

[[Migration Continued]] :  21 : 40

up() and down() methods , down is for rollback

art migrate

timestamp means  created_at and updated_at

migration : rollback will do undo of last migration command

[[To Run All Migration In One Command ]] : 34 : 40  migrate : fresh

db section artisan : 35 : 00

db : show
db : table table_name

every time you rollback last migration will undo
so fresh will undo all from begining

$table->string('country')->default('BD')->unique();

[[Default]] : 37 : 00

[[Indexing : Interesting Tips]] : 38 : 10

from the query perspective all the column you need after where , should be indexed
cause this cause apps to be slow. new migrate for doing any indexing later . Index takes time in database so be careful about that

$table ->date('birthdate');

[[Adding A New Column]] : 29 : 25

you should not edit in the schema and run migrate always unless that is new and migrate not been run

there are two ways - rollback + migrate ( by editing the schema)
or creating new migration

make : migration add_country_to_profiles

inside up() method

Schema :: table ('profiles' , function (Blueprint $table){

$table->string('country')->after('birthdate');

});

down () // for every new migration you write for custom works you need to fix the down() method to undo your last action by calling rollback

inside up() method

Schema :: table ('profiles' , function (Blueprint $table){

$table->dropColumn("country");

});

[[Column Rename]] : 32 : 40

make : migration rename_country_in_profiles

inside up() method

Schema :: table ('profiles' , function (Blueprint $table){

$table->renameColumn("Country" , "Location");

});

inside down() method

Schema :: table ('profiles' , function (Blueprint $table){

$table->renameColumn("Location" , "Country");  - doing undo of up

});

[[Indexing Two Fields]] : 42 : 20

Two person can have one phone number , so think before doing something unique

->unique('phone_number');

[[Index Command]] : 44 : 00

php artisan make : migration add_index_in_profiles

up()

$table->index('age')

down()

$table->dropIndex('age')

if you make your phone number unique then mysql will never allow to add same phone number for other person , so before validation in laravel this will work by its own

[[Status Active Inactive ]] : 45 : 38

[[enum Field]] : 46 : 00  special column that can hold double or triple possible value and default value as well

up()
$table -> enum ('status' , ['active', 'inactive'])->default('inactive');

down()

$table->dropColumn('status');

[[Dry Running Migration]] : 48 : 15

php artisan migrate --pretend

[[Nullable]] : 50 : 50

[[Foreign ID]] : 52 : 40
$table -> foreignId('user_id')->constrained(); this is new command
laravel will automatically links

1 to 1 example : Profile
1 to many example : Phone Number
many to many example : country to user (citizenship) one user can be citizen of multiple countries and in one country there could be many citizens who are many countries citizen
many to one  : phone number to user

[[Relationship]] : 53 : 00

rollback voids last migration by calling the down method of that migration

either you can delete relationships or make that null where the deleted links are recalled

[[$table->foreignId('user_id')->constrained()->onDelete('cascade') ]] : 55 : 40

[[Mostly Used Data Types]] : 59 : 00

data type mostly used : string , enum , date time , number

[[Dummy Data  / Seeding ]] : 1 : 01 : 00

this dummy data is or seeding is used for  fake data insert and sometimes faker library is used

[[make : seeder  UserSeeder]] :   1 : 02 : 40

user have some special use , as user table is laravel's default table , user-factory already have those codes to create fake data

Inside  UserSeeder

User :: factory ()->count(10)->create();

this is only for user , other custom seeding you need to write codes for dummy data

and to activate you need to go to DatabaseSeeder.php

and write $this->call([
  'UserSeeder::class',
]);

then run php artisan db : seed

[[Using Faker Library And Manually Doing User Seeding ]] : 1 : 07 : 00

fakerphp.github.io

[[Fresh Run]] : 1 : 12 : 00

Model name must be singular

php artisan migrate : refresh --seed

Question And Answer : 1 : 14 : 00

[[image seeding using faker]] : 1 : 14 : 36

[[why string for phone number storing]] : 1 : 16 : 10

due to integer limitation

[[Indexing ]] : 1 : 18 : 20

[[Big Int]]  : 1 : 21 : 30

[[Maria DB]] : 1 : 23 : 00

[[Rollback ]] : 1 : 30 : 52

will undo the last call / done things

you can jump into any near steps. like migrate : rollback --step=2

[[DB Host Vs Localhost]] : 1 : 34 : 30

[[Indexing And Making Unique]] : 1 : 35 : 00

[[How To Change Something Table Specific Without Rollback ]] : 1 : 37 : 53

[[Storing Some Text That Is Huge ]]: 1 : 39 : 17
best way is to convert them into html and then store html data into database

[[Making SuperAdmin, User And Mediator For Future Use]] : 1 : 41 : 00

[[UTF8 General CI]] : 1 : 42 : 40

while creating a database you can choose from the format they prescribed

[[Dual Index Like email And Phone Number]] : 1 : 45 : 21

$table->unique(['email', 'phone_number' ]);

[[Polymorphic Relation]] : 1 : 47 : 00

[[UTF 8 , 6 , 32 ]] : 1 : 47 : 20

By default this is UTF 8

[[Table Naming Convention]] : 1 : 48 : 30

Its better to not use factory and write codes inside seeder, but if you need to do templating and to stay in a fixed format then factory is best .

# Module-10 Live-Two-ERD Relationships And Constraints In E-Commerce Database

Date : 28 Nov 9 PM

Start : 16 : 10

[[Migration ]] : 18 : 30

For migration you need to know eloquent ORM , database and query builder eventually.
Object Relational Mapper is used as obstruction on the top of database engine to facilitates the database data as object .

[[Project Requirements]] : 29 : 00

[[Data Security Using MySql]] : 35 : 40

some validations can also be done by MySql . Moon Modeler

[[Never Send created_date() and updated_date From Front End]] : 37 : 20

timestamp code will be like that , those will be created automatically inside database

[[Can You Delete One User When Product In Add To Cart Stage]] : 37 : 36

[[Foreign Key Constraints]] : 39 : 00

Good Practice Is Doing Always :

Cascade - cascade on update
Delete - restrict on delete

[[Sample Project]] : 40 : 40

multi user login / registration like facebook , linked in
use features , category , list , sales and invoice manage

[[Phase -1 Users]] : 42 : 20

Backend api use
You have to plan in advance . when thinking about one table you need to think how many api can be generated . How the backend will work .
You have to think reversely like : api - db - backend - front end

suppose for login , registration , email verification and password reset you might find 4 api will be there like registration api , login api , password reset api , email verification api

[[Table Create]] : 48 : 00

table names will always be plural

php artisan make:migration create_users

[[Creating Timestamp]] : 52 : 30

you should create created_at and updated_at field which is timestamp

Best Way :

```
$table->timestamp('created_at')->useCurrent();
$table->timestamp('updated_at')->useCurrent()->useCurrentOnUpdate();
```

horizontal key is primary key
vertical key is unique

[[Unique]] : 55 :00

database will take this decision if you declare anything unique , then if others like laravel or any engine tries , even then data duplication is not possible

schema / model / shape / entity / table_structure / DB design all same meaning

[[php artisan migrate ]] : 58 : 20

[[Behavior Check After Schema Design]] 1 : 00 : 00

[[Phase 2 Categories]] : 1 : 04 : 00

[[Foreign Key]] : 1 : 11 : 20

Categories_user_id_foreign
1 To Many : One user can have many categories
if there is no user then how come categories will exist . so first you need to build users then build categories

so while making foreign key user_id in the categories table you need to know that both data types must have to be same : users table -> id categories table -> user_id

[[Common Rules & Shortcut To Rememeber]] : 1 : 14 : 00

Foreign Key : পাত্র

Primary Key : পাত্রী
ON ('user_id') : পাত্রীর বাসা

$table -> unsignedBigInteger('user_id');
$table->foreign('user_id')->reference('id')->on('users');

you can add two rules :

$table->foreign('user_id')->reference('id')->on('users')->restrictOnDelete()->cascadeOnUpdate();

[[->restrictOnDelete()->cascadeOnUpdate()]] : 1 : 18 : 30

[[Clickable Retain]] : 1 : 21 : 00

[[Check If Delete & Update Possible ]] : 1 : 23 : 00

[[Phase 3 Customers]] : 1 : 28 : 00

[[Phase 4 Products]] : 1 : 31 : 00

so one user can have many categories and one product also can have many categories

inside products table there will be two foreign keys

$table->unsignedBigInt('user_id');
$table->unsignedBigInt('category_id');

$table->foreign('user_id')->reference('id')->on('users')->restrictOnDelete()->cascadeOnUpdate();

$table->foreign('category_id')->reference('id')->on('categories')->restrictOnDelete()->cascadeOnUpdate();

[[Never insert images In Database ]] : 1 : 35 : 00

[[Never Create Migration File Manually In PhpMyAdmin]] : 1 : 37 : 00

[[Phase 5 Invoices]] : 1 : 38 : 00

In this sample project user is the seller

one customer can have many invoices same way one user can have many invoices
first you need to create column and then create relation

so the invoices table will hold two foreign key like before
customer_id and user_id
unsigned big integer

[[Relationship Like Before]] : 1 : 42 : 00

we now could track the user and customer related with the users but keep in mind there is another item which is very important : that is product .

Invoice have many to many relationship with product

one invoice can have many products and also one product can be in many invoices

so two one to many can create one many to many

[[Invoice_Products Table]] : 1 : 50 : 00

php artisan make : migration create_invoice_products

naming convention : naming should be done alphabetically and also if both first letter is of same alphabet , second one should be considered

this is known as pivot table sometimes to link everyone from one place

if you follow the naming convention laravel will detect tables automatically , even when model will created

model name is always singular invoiceProduct (model will be like this for this table )

created_at and updated_at are always needed for report generation

[[Invoice_Products_Continued ]] : 2 : 00 : 00

user_id
invoice_id
product_id

[[ERD Diagram ]] : 2 : 03 : 00

Question And Answer : 2 : 06 : 00

[[PHP Vulnerability]] : 2 : 07 : 00

[[High Traffic Order , Stock Holding ]]  : 2 : 08 : 00

[[Time Stamp Is A Great Tool To Hold Data ]] : 2 : 12 : 32

time stamp can hold data up to mili seconds, in that case which buyer ordered first can easily be identified, so heavy traffic wont be a problem

[[db forge ]] : 2 : 15 : 17

db forge is not for always. so you can work with the trial version

[[length 50]] : 2 : 16 : 40

[[obsidian]] : 2 : 18 : 00

there is no problem putting cascade and restrict in any order .

[[ERD Diagram]] : 2 : 21 : 14

[[Junction Table Or Pivot Table]] : 2 : 25 : 10

[[Big Data]] : 2 : 26 : 00

Big data have volume , velocity , variety , value and veracity
Single data have only volume

Business application is generally simple data

[[Using Junction Reduces Query Complexity]] : 2 : 29 : 15

some times they do not use junction or pivot table , rather they use temp table

some times data comes from view not any table then junction table is important

[[Flag Raise ]] : 2 : 30 : 52

Redis , session , database , php session is useful

Complex - Redis is good and keep database intact
product stable

[[Markdown]] : 2 : 34 : 10

[[Avoid Complex Thought Rather Think How Make Easy Solve]] : 2 : 36 : 30

web application development is easy

Database / MySql is the final destination and fixed and later can not be changed in any way

[[Database Design]] : 2 : 41 : 55

So first DB then backend and then plan your front end accordingly

SRS / BRD / SRA
Software Requirement Analysis
Business Requirement Documentation
Software Requirement Specification

[[SRS]]  : 2 : 43 : 00

[[Auto Indexing]] : 2 : 47 : 40

[[Index And View Later Will Be Discussed]] : 2 : 50 : 10

[[Buffet Job Circular]] : 3 : 00 : 00

# Module-10 Conceptual-One- Laravel Migration And Relationships Music App Database Schema Design

Date : 29 Nov 9 PM

Start : 20:00

[[SRS]] : 22 :00

[[Relation Breakdown]] : 28 : 00

[[Pivot Table]] : 32 : 00

this is also called intermediate level table

playlist_tracks ( many to many )

two one to many combined into one many to many

[[Coding Of Tables]] : 43 : 50

->constrained() automatically detects foreignId

[[Many To Many Relation]] : 1 : 07 : 00

[[Naming Convention ]] : 1 : 07 : 50

Question & Answer  1 : 16 : 20

[[Many To Many Intermediate Pivot Table ]] : 1 : 16 : 40

[[Table Plus Database ]] : 1 : 20 : 00

table plus is db  client  and free just like postman

db forge substitute is dbgate

[[Why no cascade delete and restrict update]] : 1 : 22 : 00

[[->constrained is new technology]] : 1 : 25 : 45

[[Which Table After Which Table]] : 1 : 29 : 50

[[Draw SQL]] : 1 : 33 : 10

[[Naming Convention Helps While Using Model]] : 1 : 39 :00