

ព្រះរាជាណាចក្រកម្ពុជា
ជាតិ សាសនា ព្រះមហាក្សត្រ



សាកលវិទ្យាល័យ បៀលប្រាយ
(ខេត្តសៀមរាប)

មហាវិទ្យាល័យ
វិទ្យាសាស្ត្រ និងបច្ចេកវិទ្យា

ជំនាញ៖ បច្ចេកវិទ្យាព័ត៌មាន
ប្រភេទកិច្ចការ៖ កិច្ចការស្រាវជ្រាវជាក្រុម
មុខវិជ្ជា៖ Advance PHP and MySQL
ប្រធានបទ៖ ប្រព័ន្ធគ្រប់គ្រងភោជនីយដ្ឋាន (Restaurant Management System)
សាស្ត្រាចារ្យណែនាំ៖ **ម៉ុន មិញ**

ក្រុមទី ២

១. លន់ សុវណ្ណារ

២. អែម សុផែ

៣. ឡឺន ឡេងយ៉ូ

៤. ម៉ាន ឈាងម៉ែ

៥. តីក សីងហេង

៦. ស្មីត ឡុងឌី

៧. ផុល សុផាវ៉ា

ជំនាន់ទី១៨ ឆ្នាំទី៣ ឆមាសទី១ ក្រុមD1IT បន្ទប់D203 វេនយប់ ឆ្នាំសិក្សា ២០២១-២០២២

កិច្ចការ		បទបង្ហាញ (Online)		LLO	CLO
ចំណុចវាយតម្លៃ	ពិន្ទុទទួលបាន	ចំណុចវាយតម្លៃ	ពិន្ទុទទួលបាន		
ខ្លឹមសារ ៨០%		បង្ហាញលទ្ធផល ៤០%		LL1,	CLO1
ទម្រង់ ១០%		បកស្រាយ ៥០%		LLO2	
អក្ខរាវិរុទ្ធ ១០%		កាយវិការ ១០%		And	
សរុប ១០០%	%	សរុប ១០០%	%	LLO6	

មាតិកា



ចំណងជើង

ទំព័រ

I. រំលឹកមេរៀន.....	1
1.1. មេរៀនទី១ ចាប់ផ្តើមជាមួយ Laravel.....	1
1.1.1. អ្វីទៅជា Laravel Framework	1
1.1.2. ការបង្កើត Laravel Project.....	1
1.2. មេរៀនទី២ Route និង Controller	10
1.2.1. អ្វីជា MVC	10
1.2.2. អំពី Route	10
1.2.3. អំពី Controller	12
1.2.4. អំពី Request និង Response.....	17
1.3. មេរៀនទី៣ View និង Blade	20
1.3.1. អ្វីជា View.....	20
1.3.2. អ្វីជា Blade Template	20
1.3.3. អ្វីជា Form Handling	20
1.3.4. Validation.....	22
1.4. មេរៀនទី៤ Database និង Eloquent.....	25
1.4.1. សេចក្តីផ្តើម	25
1.4.2. Configuration.....	25
1.4.3. Query Builder.....	30
1.4.4. Eloquent.....	33
1.4.5. Relation	37
II. Restaurant Management System	39
2.1. Source Code (Github)	39
2.2. User Interface	39
2.2.1. Login Page.....	39
2.2.2. Register Page.....	39
2.2.3. Forgot Password Page.....	39
2.2.4. Reset Password Page.....	40
2.2.5. Dashboard Page.....	40
2.2.6. Order Detail Page.....	40
2.2.6.1. Create Modal	41

2.2.6.2. View Modal.....	41
2.2.6.3. Update Modal.....	41
2.2.6.4. Delete Modal.....	41
2.2.7. Order Page	42
2.2.6.1. Create Modal	42
2.2.6.2. View Modal.....	42
2.2.6.3. Update Modal.....	42
2.2.8. Item Page.....	43
2.2.6.1. Create Modal	43
2.2.6.2. View Modal.....	43
2.2.6.3. Update Modal.....	44
2.2.9. Table Page.....	45
2.2.6.1. Create Modal	45
2.2.6.2. View Modal.....	45
2.2.6.3. Update Modal.....	45
2.2.10. Category Page.....	46
2.2.6.1. Create Modal	46
2.2.6.2. View Modal.....	46
2.2.6.3. Update Modal.....	47
2.2.11. Employee Page.....	47
2.2.6.1. Create Modal	47
2.2.6.2. View Modal.....	48
2.2.6.3. Update Modal.....	49

I. វិចិត្រកមេរៀន

មេរៀនទី១ ការចាប់ផ្តើមជាមួយ Laravel

1. អ្វីទៅជា Laravel Framework

Laravel framework គឺជាPHP web framework កូដចំហរនិងឥតគិតថ្លៃដែលត្រូវបានបង្កើតឡើងដោយលោក Taylor Otwell និងមានបំណងសម្រាប់ការបង្កើតកម្មវិធី (web application) តាមគំរូស្ថាបត្យកម្ម Model-view-Controller) MVC) និងផ្អែកលើ symfony ។

2. ការបង្កើត Laravel Project

ដើម្បីអាចបង្កើតlaravel project បានយើងចាំបាច់ត្រូវការកម្មវិធីមួយជាចាំបាច់គឺcomposer (<https://getcomposer.org/Composer-Setup.exe>)ដែលComposer គឺជា Tool មួយ សំរាប់ការរៀបចំ Package ឬ Libraries ឲ្យមានភាពអាស្រ័យទៅវិញទៅមក ដែលអនុញ្ញាតឲ្យអ្នកប្រើប្រាស់ install ឬ update ក្នុង Project នៅក្នុង PHP ។ ឧបមាថា ក្នុង Project របស់អ្នក ពឹងផ្អែកលើ Library មួយចំនួន ហើយ ក្នុងចំណោម Library ទាំងនេះ ពឹងផ្អែកទៅលើ Library ដទៃទៀតដែរ ។ ដូច្នេះ ការប្រើប្រាស់ Composer នឹងផ្តល់ ភាពងាយស្រួល ក្នុងការទាញយក Library ដែលត្រូវការ ស្របទៅតាម Version នៅក្នុង Project នោះ។វាត្រូវបានគេប្រើប្រាស់គ្រប់ប្រភេទ PHP Framework ដូចជា Symfony, Laravel ជាដើម ។

laravelប្រើប្រាស់ composer ដើម្បីគ្រប់គ្រងរាល់ dependenciesទាំងអស់របស់វា។ដូច្នេះមុននឹងប្រើlaravelត្រូវប្រាកដថាcomposer ត្រូវបានតំឡើងត្រឹមត្រូវ។

+ តម្រូវការប្រព័ន្ធ

ដើម្បីអោយ laravel application តំណើរការបានល្អយើងត្រូវការនូវសាមាសធាតុសំខាន់ៗ មួយចំនួនដូចខាងក្រោម៖

- PHP >= 7.3 ឡើងទៅ
- OpenSSL PHP Extension
- PDO PHP Extension

- Tokenizer PHP Extension
- XML PHP Extension
- JSON PHP Extension

+ ការបង្កើតProject

គេមានវិធីសាស្ត្រចំនួនពីរដើម្បីបង្កើត laravel project គឺដោយប្រើប្រាស់នូវ composer command និង laravel command។

+ ការបង្កើតដោយប្រើ laravel command

1. ដាក់ប្លង់ត្រូវដំឡើង laravel command ជាមុនសិនតាមរយៈបញ្ជាខាងក្រោម
 - `composer global require laravel/installer`
2. បន្ទាប់មកទៀតត្រូវប្រើនូវបញ្ជាខាងក្រោមដើម្បីបង្កើត project
 - `laravel new <name_of_project>`
3. ហើយpress enter key
4. រង់ចាំរហូតដល់វាដោនឡូតចប់

+ ការបង្កើតដោយប្រើនូវ composer command

1. `Composer create-project --prefer-dist laravel/laravel <project_name>`
2. ហើយpress enter key
3. រង់ចាំរហូតដល់វាដោនឡូតចប់

សំគាល់៖ កំឡុងដែលបង្កើត project ចាំបាច់ត្រូវតែមាន internet។

ឧទាហរណ៍៖ បង្កើត laravel project មួយមានឈ្មោះថា blog

1. ជ្រើសរើសទីតាំងដើម្បីបង្កើត project
2. បង្កើតថតឯកសារមួយដើម្បីដាក់ project → រួចបើកថតឯកសារដែលបានបង្កើតរួច
3. ចុច shift + right click → open PowerShell windows here
4. បន្ទាប់មកទៀតវាយបញ្ជា *`composer global require laravel/installer`*
5. ក្រោយពីវាតំណើរការចប់ យើងបង្កើត laravel project តាមបញ្ជាខាងក្រោម
 - `c:/> laravel new blog`
6. រង់ចាំរហូតដល់វាដោនឡូតចប់

```

chheangmai@chheangmai-koompi$~ composer global require laravel/installer
Changed current directory to /home/chheangmai/.config/composer
Using version ^4.2 for laravel/installer
./composer.json has been created
Running composer update laravel/installer
Loading composer repositories with package information
Updating dependencies
Lock file operations: 10 installs, 0 updates, 0 removals
  - Locking laravel/installer (v4.2.10)
  - Locking psr/container (2.0.2)
  - Locking symfony/console (v6.0.3)
  - Locking symfony/polyfill-ctype (v1.24.0)
  - Locking symfony/polyfill-intl-grapheme (v1.24.0)
  - Locking symfony/polyfill-intl-normalizer (v1.24.0)
  - Locking symfony/polyfill-mbstring (v1.24.0)
  - Locking symfony/process (v6.0.3)
  - Locking symfony/service-contracts (v3.0.0)
  - Locking symfony/string (v6.0.3)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 10 installs, 0 updates, 0 removals
  - Downloading symfony/process (v6.0.3)
  - Downloading symfony/polyfill-mbstring (v1.24.0)
  - Downloading symfony/polyfill-intl-normalizer (v1.24.0)
  - Downloading symfony/polyfill-intl-grapheme (v1.24.0)
  - Downloading symfony/polyfill-ctype (v1.24.0)
  - Downloading symfony/string (v6.0.3)
  - Downloading psr/container (2.0.2)
  - Downloading symfony/service-contracts (v3.0.0)
  - Downloading symfony/console (v6.0.3)
  - Downloading laravel/installer (v4.2.10)
  - Installing symfony/process (v6.0.3): Extracting archive
  - Installing symfony/polyfill-mbstring (v1.24.0): Extracting archive
  - Installing symfony/polyfill-intl-normalizer (v1.24.0): Extracting archive
  - Installing symfony/polyfill-intl-grapheme (v1.24.0): Extracting archive
  - Installing symfony/polyfill-ctype (v1.24.0): Extracting archive
  - Installing symfony/string (v6.0.3): Extracting archive
  - Installing psr/container (2.0.2): Extracting archive
  - Installing symfony/service-contracts (v3.0.0): Extracting archive
  - Installing symfony/console (v6.0.3): Extracting archive
  - Installing laravel/installer (v4.2.10): Extracting archive
6 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating autoload files
8 packages you are using are looking for funding.

```



```
Package operations: 10 installs, 0 updates, 0 removals
- Downloading symfony/process (v6.0.3)
- Downloading symfony/polyfill-mbstring (v1.24.0)
- Downloading symfony/polyfill-intl-normalizer (v1.24.0)
- Downloading symfony/polyfill-intl-grapheme (v1.24.0)
- Downloading symfony/polyfill-ctype (v1.24.0)
- Downloading symfony/string (v6.0.3)
- Downloading psr/container (2.0.2)
- Downloading symfony/service-contracts (v3.0.0)
- Downloading symfony/console (v6.0.3)
- Downloading laravel/installer (v4.2.10)
- Installing symfony/process (v6.0.3): Extracting archive
- Installing symfony/polyfill-mbstring (v1.24.0): Extracting archive
- Installing symfony/polyfill-intl-normalizer (v1.24.0): Extracting archive
- Installing symfony/polyfill-intl-grapheme (v1.24.0): Extracting archive
- Installing symfony/polyfill-ctype (v1.24.0): Extracting archive
- Installing symfony/string (v6.0.3): Extracting archive
- Installing psr/container (2.0.2): Extracting archive
- Installing symfony/service-contracts (v3.0.0): Extracting archive
- Installing symfony/console (v6.0.3): Extracting archive
- Installing laravel/installer (v4.2.10): Extracting archive
6 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating autoload files
8 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
chheangmai@chheangmai-koompi$~
```



```
- Locking league/mime-type-detection (1.9.0)
- Locking mockery/mockery (1.5.0)
- Locking monolog/monolog (2.3.5)
- Locking myclabs/deep-copy (1.10.2)
- Locking nesbot/carbon (2.56.0)
- Locking nette/schema (v1.2.2)
- Locking nette/utils (v3.2.7)
- Locking nikic/php-parser (v4.13.2)
- Locking nunomaduro/collision (v6.1.0)
- Locking phar-io/manifest (2.0.3)
- Locking phar-io/version (3.1.1)
- Locking phpdocumentor/reflection-common (2.2.0)
- Locking phpdocumentor/reflection-docblock (5.3.0)
- Locking phpdocumentor/type-resolver (1.6.0)
- Locking phpoption/phpooption (1.8.1)
- Locking phpspec/prophecy (v1.15.0)
- Locking phpunit/php-code-coverage (9.2.10)
- Locking phpunit/php-file-iterator (3.0.6)
- Locking phpunit/php-invoker (3.1.1)
- Locking phpunit/php-text-template (2.0.4)
- Locking phpunit/php-timer (5.0.3)
- Locking phpunit/phpunit (9.5.13)
- Locking psr/container (2.0.2)
- Locking psr/event-dispatcher (1.0.0)
- Locking psr/http-client (1.0.1)
- Locking psr/http-factory (1.0.1)
- Locking psr/http-message (1.0.1)
- Locking psr/log (3.0.0)
- Locking psr/simple-cache (3.0.0)
- Locking psy/psysh (v0.11.1)
- Locking ralouphie/getallheaders (3.0.3)
- Locking ramsey/collection (1.2.2)
- Locking ramsey/uuid (4.2.3)
- Locking sebastian/cli-parser (1.0.1)
- Locking sebastian/code-unit (1.0.8)
- Locking sebastian/code-unit-reverse-lookup (2.0.3)
- Locking sebastian/comparator (4.0.6)
- Locking sebastian/complexity (2.0.2)
- Locking sebastian/diff (4.0.4)
- Locking sebastian/environment (5.1.3)
- Locking sebastian/exporter (4.0.4)
- Locking sebastian/global-state (5.0.4)
- Locking sebastian/lines-of-code (1.0.3)
- Locking sebastian/object-enumerator (4.0.4)
```

```
- Locking symfony/polyfill-intl-grapheme (v1.24.0)
- Locking symfony/polyfill-intl-idn (v1.24.0)
- Locking symfony/polyfill-intl-normalizer (v1.24.0)
- Locking symfony/polyfill-mbstring (v1.24.0)
- Locking symfony/polyfill-php72 (v1.24.0)
- Locking symfony/polyfill-php80 (v1.24.0)
- Locking symfony/polyfill-php81 (v1.24.0)
- Locking symfony/process (v6.0.3)
- Locking symfony/routing (v6.0.3)
- Locking symfony/service-contracts (v3.0.0)
- Locking symfony/string (v6.0.3)
- Locking symfony/translation (v6.0.3)
- Locking symfony/translation-contracts (v3.0.0)
- Locking symfony/var-dumper (v6.0.3)
- Locking theseer/tokenizer (1.2.1)
- Locking tijsverkoyen/css-to-inline-styles (2.2.4)
- Locking vlucas/phpdotenv (v5.4.1)
- Locking voku/portable-ascii (2.0.0)
- Locking webmozart/assert (1.10.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 109 installs, 0 updates, 0 removals
- Downloading doctrine/inflector (2.0.4)
- Downloading doctrine/lexer (1.2.2)
- Downloading symfony/polyfill-ctype (v1.24.0)
- Downloading webmozart/assert (1.10.0)
- Downloading dragonmantank/cron-expression (v3.3.1)
- Downloading symfony/deprecation-contracts (v3.0.0)
- Downloading psr/container (2.0.2)
- Downloading fakerphp/faker (v1.19.0)
- Downloading symfony/polyfill-mbstring (v1.24.0)
- Downloading symfony/http-foundation (v6.0.3)
- Downloading psr/event-dispatcher (1.0.0)
- Downloading symfony/event-dispatcher-contracts (v3.0.0)
- Downloading symfony/event-dispatcher (v6.0.3)
- Downloading symfony/var-dumper (v6.0.3)
- Downloading psr/log (3.0.0)
- Downloading symfony/error-handler (v6.0.3)
- Downloading symfony/http-kernel (v6.0.4)
- Downloading voku/portable-ascii (2.0.0)
- Downloading symfony/polyfill-php80 (v1.24.0)
- Downloading phpoption/phpooption (1.8.1)
- Downloading graham-campbell/result-type (v1.0.4)
- Downloading vlucas/phpdotenv (v5.4.1)
```

```
- Installing doctrine/inflector (2.0.4): Extracting archive
- Installing doctrine/lexer (1.2.2): Extracting archive
- Installing symfony/polyfill-ctype (v1.24.0): Extracting archive
- Installing webmozart/assert (1.10.0): Extracting archive
- Installing dragonmantank/cron-expression (v3.3.1): Extracting archive
- Installing symfony/deprecation-contracts (v3.0.0): Extracting archive
- Installing psr/container (2.0.2): Extracting archive
- Installing fakerphp/faker (v1.19.0): Extracting archive
- Installing symfony/polyfill-mbstring (v1.24.0): Extracting archive
- Installing symfony/http-foundation (v6.0.3): Extracting archive
- Installing psr/event-dispatcher (1.0.0): Extracting archive
- Installing symfony/event-dispatcher-contracts (v3.0.0): Extracting archive
- Installing symfony/event-dispatcher (v6.0.3): Extracting archive
- Installing symfony/var-dumper (v6.0.3): Extracting archive
- Installing psr/log (3.0.0): Extracting archive
- Installing symfony/error-handler (v6.0.3): Extracting archive
- Installing symfony/http-kernel (v6.0.4): Extracting archive
- Installing voku/portable-ascii (2.0.0): Extracting archive
- Installing symfony/polyfill-php80 (v1.24.0): Extracting archive
- Installing phpoption/phpooption (1.8.1): Extracting archive
- Installing graham-campbell/result-type (v1.0.4): Extracting archive
- Installing vlucas/phpdotenv (v5.4.1): Extracting archive
- Installing symfony/css-selector (v6.0.3): Extracting archive
- Installing tijsverkoyen/css-to-inline-styles (2.2.4): Extracting archive
- Installing symfony/routing (v6.0.3): Extracting archive
- Installing symfony/process (v6.0.3): Extracting archive
- Installing symfony/polyfill-php72 (v1.24.0): Extracting archive
- Installing symfony/polyfill-intl-normalizer (v1.24.0): Extracting archive
- Installing symfony/polyfill-intl-idn (v1.24.0): Extracting archive
- Installing symfony/mime (v6.0.3): Extracting archive
- Installing symfony/service-contracts (v3.0.0): Extracting archive
- Installing egulias/email-validator (3.1.2): Extracting archive
- Installing symfony/serializer (v6.0.3): Extracting archive
- Installing symfony/finder (v6.0.3): Extracting archive
- Installing symfony/polyfill-intl-grapheme (v1.24.0): Extracting archive
- Installing symfony/string (v6.0.3): Extracting archive
- Installing symfony/console (v6.0.3): Extracting archive
- Installing symfony/polyfill-php81 (v1.24.0): Extracting archive
- Installing ramsey/collection (1.2.2): Extracting archive
- Installing brick/math (0.9.3): Extracting archive
- Installing ramsey/uuid (4.2.3): Extracting archive
- Installing psr/simple-cache (3.0.0): Extracting archive
- Installing symfony/translation-contracts (v3.0.0): Extracting archive
- Installing symfony/translation (v6.0.3): Extracting archive
```

```
- Installing sebastian/comparator (4.0.6): Extracting archive
- Installing sebastian/code-unit (1.0.8): Extracting archive
- Installing sebastian/cli-parser (1.0.1): Extracting archive
- Installing phpunit/php-timer (5.0.3): Extracting archive
- Installing phpunit/php-text-template (2.0.4): Extracting archive
- Installing phpunit/php-invoker (3.1.1): Extracting archive
- Installing phpunit/php-file-iterator (3.0.6): Extracting archive
- Installing theseer/tokenizer (1.2.1): Extracting archive
- Installing sebastian/lines-of-code (1.0.3): Extracting archive
- Installing sebastian/complexity (2.0.2): Extracting archive
- Installing sebastian/code-unit-reverse-lookup (2.0.3): Extracting archive
- Installing phpunit/php-code-coverage (9.2.10): Extracting archive
- Installing doctrine/instantiator (1.4.0): Extracting archive
- Installing phpspec/prophecy (v1.15.0): Extracting archive
- Installing phar-io/version (3.1.1): Extracting archive
- Installing phar-io/manifest (2.0.3): Extracting archive
- Installing myclabs/deep-copy (1.10.2): Extracting archive
- Installing phpunit/phpunit (9.5.13): Extracting archive
- Installing spatie/backtrace (1.2.1): Extracting archive
- Installing spatie/flare-client-php (1.0.1): Extracting archive
- Installing spatie/ignition (1.0.2): Extracting archive
- Installing spatie/laravel-ignition (1.0.4): Extracting archive
73 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
Discovered Package: fruitcake/laravel-cors
Discovered Package: laravel/sail
Discovered Package: laravel/sanctum
Discovered Package: laravel/tinker
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: spatie/laravel-ignition
Package manifest generated successfully.
78 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
No publishable resources for tag [laravel-assets].
Publishing complete.
> @php artisan key:generate --ansi
Application key set successfully.

Application ready! Build something amazing.
chheangmai@chheangmai-koompi$~
```

មេរៀនទី២ ROUTE និង CONTROLLER

2.1. អ្វីជា MVC?

ជា software design pattern ដែលវាធ្វើការបំបែក application ជា ៣ផ្នែកគឺ model, view និង controller ដើម្បីបង្កើតភាពងាយស្រួលដល់ developer ដើម្បី maintenance application ។

- Model (is a data and business logic) ជាការបង្ហាញទំរង់នៃទិន្នន័យនិង business logic. វាជាអ្នកគ្រប់គ្រងទិន្នន័យរបស់កម្មវិធីដែលមានតួនាទីជាអ្នកទទួលនិងរក្សាទុកទិន្នន័យក្នុងជាតាបេស។
- View (is a User Interface) ជាអ្វីៗដែលអ្នកទស្សនាមើលឃើញដែលវាបង្ហាញទិន្នន័យដោយប្រើប្រាស់នូវ model ទៅកាន់អ្នកទស្សនានិងអោយគេធ្វើការកែប្រែទិន្នន័យបាន។
- Controller (request handler) ជាអ្នកដោះស្រាយនូវសំណើរបស់អ្នកប្រើប្រាស់ (user) ។ ជាទូទៅអ្នកប្រើប្រាស់ធ្វើអន្តរកម្មជាមួយ View តាមរយៈការលើកជាសំណើរ URL ត្រឹមត្រូវដែលសំណើរទាំងត្រូវបានដោះស្រាយដោយ Controller ។ បន្ទាប់មក controller នឹងបង្ហាញនូវចំណែកតាមរយៈ data model ទៅជា view ។

2.2. អំពី Route ?

Route ប្រៀបបាននិងផ្លូវឬផែនទីដែលគេប្រើប្រាស់សម្រាប់បង្កើត user's URL request ។ Route គឺវាមានសារៈសំខាន់ណាស់សំរាប់ Laravel framework ពីព្រោះវាជាអ្នកកំណត់ថាតើត្រូវបង្ហាញ Page មួយណានៅពេលដែលយើងធ្វើ request ចឹងហើយ Route គឺជាអ្នកកំណត់ហើយទីតាំងរបស់ resource ។

- ការបង្កើត Route ដើម្បីបង្កើត user's request បានចំបាច់ត្រូវបង្កើត route ជាមុនសិន។ ដូច្នេះដើម្បីបង្កើត route ត្រូវប្រើប្រាស់នូវ Route class façade ដូចខាងក្រោម៖

Syntax:

Route::method(\$url,\$callback);

- method ជារបៀបនៃការបញ្ជូន request
- \$url ជាបណ្តុំនៃ user's request (URL pattern ឬ endpoints) ដែលវាខ័ណ្ឌគ្នាដោយសញ្ញា (/) ។
- \$callback ជា function ដែលត្រូវ execute នូវ request (user) ហើយបង្កើតបាន response (server) ។

ឧទាហរណ៍៖

```

1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  Route::get("/example", function () {
6      return 'this is my first route in laravel';
7  });
8
9
10

```

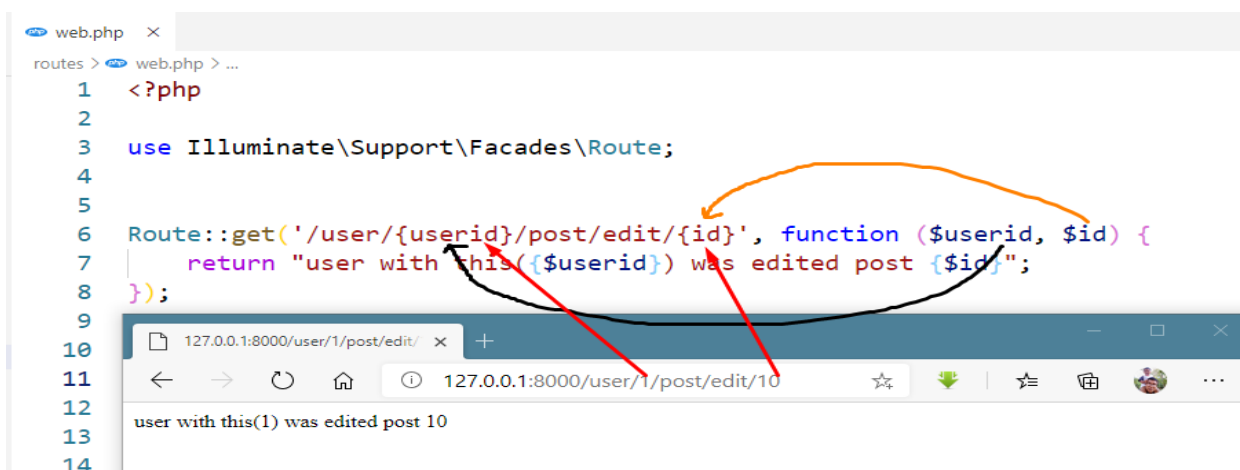
Method uri

callback function

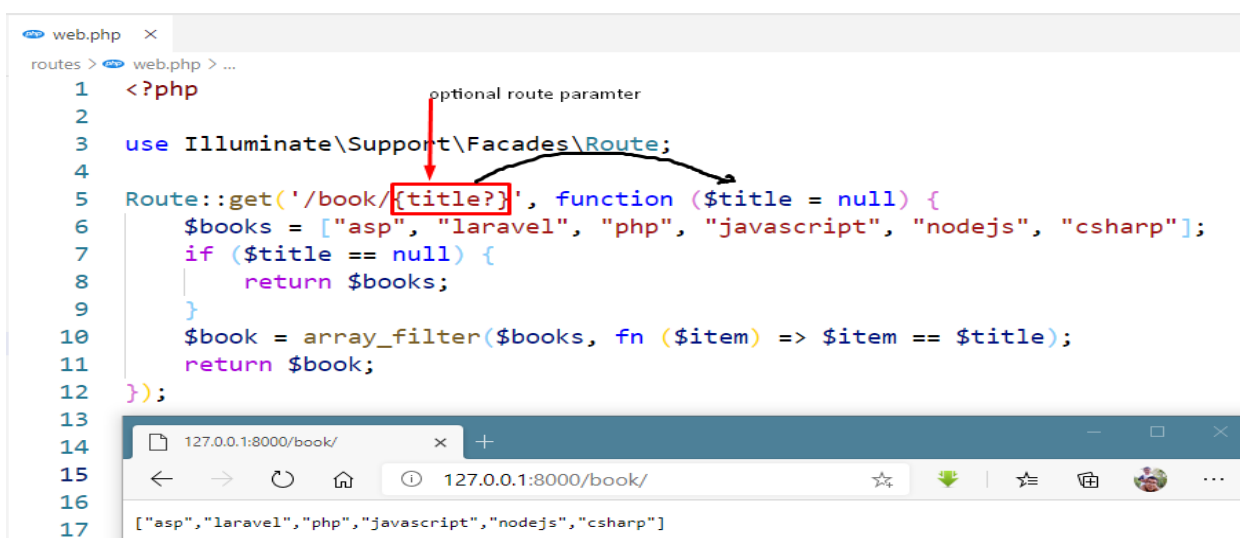
- Route Parameters

route parameters ត្រូវតែស្ថិតនៅក្នុងសញ្ញា {parameter} ដែលគេចែកជាពីរ គឺ Required parameters និង optional parameters ហើយវាខុសគ្នាត្រង់ថាមួយមានសញ្ញាស(?) ពីក្រោយនិងមួយទៀតអត់មាន។

Require Route Parameter



Optional Route Parameter



វាជាrouteប៉ារ៉ាម៉ែត្រមួយប្រភេទដែលផ្តល់តំលៃអោយក៏បានមិនអោយក៏បាន។ ដើម្បីបញ្ជាក់ថាវាជាប្រភេទ optional parameterគឺត្រូវមានសញ្ញាសួរនៅពីក្រោយparameter(?)។

-ការដាក់ឈ្មោះអោយ Route

ការដាក់ឈ្មោះអោយrouteផ្តល់ភាពងាយស្រួលក្នុងការបង្កើត

URLsឬក៏ផ្លាស់ប្តូរទិសដៅ(redirect)សម្រាប់routeដាក់លាក់ណាមួយ។យើងអាចដាក់ឈ្មោះអោយrouteដោយប្រើ នូវ name() method ដោយគ្រាន់តែចង(changing)ភ្ជាប់វាជាមួយនិង route របស់យើង។

-Route Constrains

Route constrain ជាគោលការណ៍ក្នុងការកំណត់ពីប្រភេទតំលៃរបស់user's

requestថាតើវាបានផ្ទៀងផ្ទាត់ជាមួយនិងroute's parameterឬអត់។ជាមួយ laravel framework

ដើម្បីកំណត់នូវroute constrain គឺគេប្រើ where() ជាមួយ regular expressionឬក៏ global helper methodដូចជា៖ whereNumber()និង whereAlpha()។

- whereNumber(\$parameters) តំលៃរបស់route អាចផ្តល់បានតែលេខ
- whereAlpha(\$parameters) តំលៃរបស់route អាចផ្តល់បានតែអក្សរ

ឧទាហរណ៍៖

```
web.php x
routes > web.php > ...
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5
6  Route::get("/{id}", function ($id) {
7      return $id;
8  })->where('id', '[0-9]+');
9  //-----
10 Route::get("/edit/{name}", function ($name) {
11     return $name;
12 })->where('name', '[a-zA-Z]+');
13 //-----
14 Route::get('post/{role}/{id}', function ($role, $id) {
15     return "role:{$role},id:{$id}";
16 })->where(['role' => '[a-z]+', 'id' => '[0-9]+']);
```

2.3. អំពី Controller

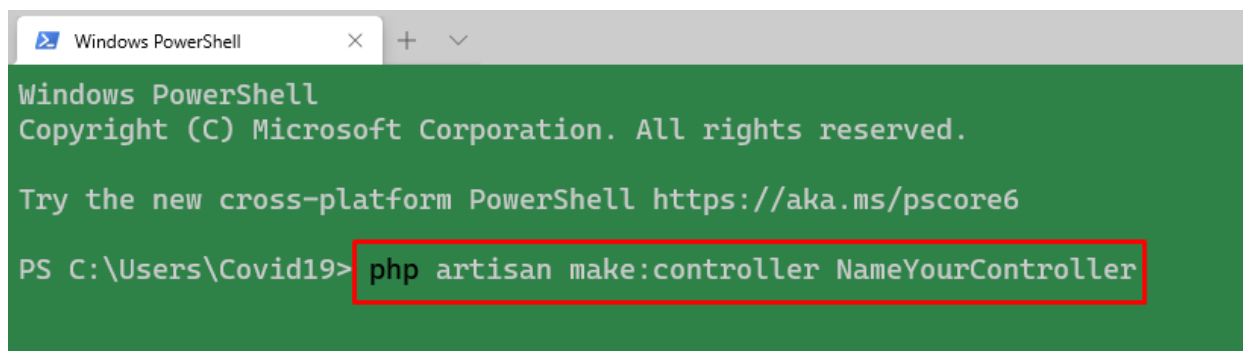
Controller គឺជាphp class ដែលដាក់ជាក្រុមនូវ associated request handling logic ក្នុងfile

តែមួយហើយវាប្រមូលផ្តុំ public methods(functions)ជាច្រើនដែលមួយ method

ទទួលខុសត្រូវលើសំណើរមួយ(request)។controller class ត្រូវបានរក្សាទុកនៅក្នុងថត app\Http\Controllersនៃ laravel project។

2.3.1. ការបង្កើត Controller

Controller ត្រូវបានបង្កើតដោយបញ្ជាខាងក្រោម៖



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

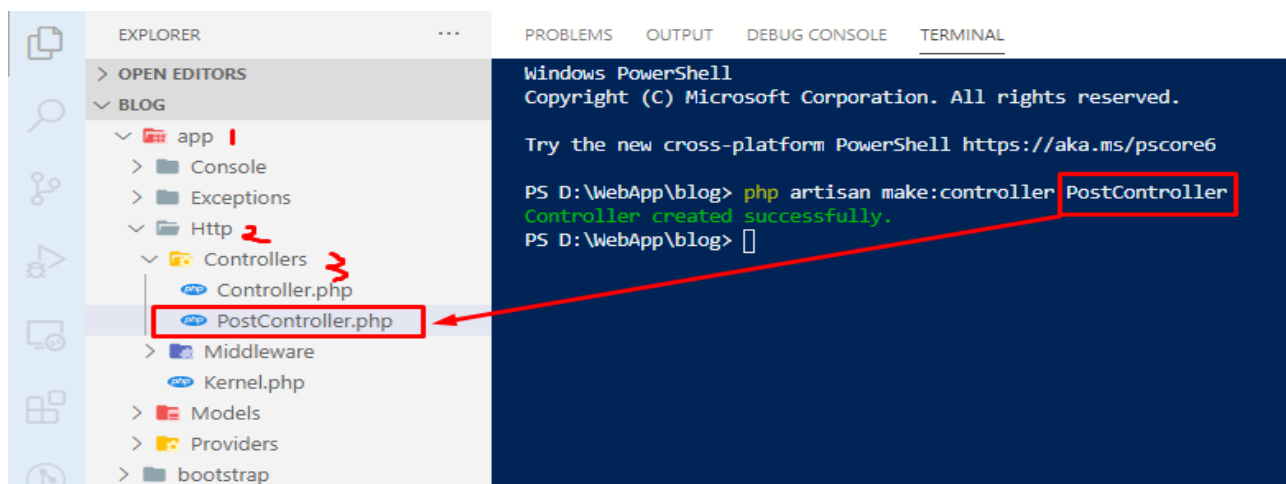
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Covid19> php artisan make:controller NameYourController
```

ចំណាំ៖

- ដើម្បីប្រើបញ្ជា php artisan បានចាំបាច់ត្រូវចូលទៅ laravel project ជាមុនសិន
- ឈ្មោះរបស់ controller ត្រូវផ្ដើមដោយអក្សរធំជានិច្ច(Pascal case)
- ឈ្មោះរបស់ controller ត្រូវបញ្ចប់ដោយពាក្យ Controller
- ឈ្មោះរបស់មិនចាប់ផ្ដើមដោយសញ្ញាពិសេសទេ៖ \$, &, *, #, @ជាដើម។

ឧទាហរណ៍៖



ក្រោយពីបង្កើត controller មួយដែលមានទំរង់ដូចខាងក្រោម៖

```
PostController.php X
app > Http > Controllers > PostController.php > PostController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class PostController extends Controller
8  {
9      //you will define your public methods here
10 }
```

បន្ទាប់មកធ្វើការបង្កើត public method នៅក្នុងcontroller

```
PostController.php X
app > Http > Controllers > PostController.php > PostController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class PostController extends Controller
8  {
9      public function example()
10     {
11         return "this is first public method in controller.";
12     }
13
14     public function example2()
15     {
16         return "this is second public method in controller.";
17     }
18 }
```

- Routing Controller

Controller's public methods

មិនត្រូវបានតំណើរការដោយស្វ័យប្រវត្តិទេយើងត្រូវតែបង្កើតrouteជាមុនសិនតាមរូបមន្តដូចខាងក្រោម៖

`Route::method($url,[ControllerName::class,methodName]);`

- ControllerName ជាឈ្មោះរបស់ controllerដែលយើងបានបង្កើត
- methodName ជាឈ្មោះរបស់method ដែលមាននៅក្នុងcontrollerដែលបានបង្កើត។

ចំណាំ៖

ដើម្បីប្រើប្រាស់controller class បានយើងត្រូវតែនាំយកcontroller namespaceមកដាក់ផ្នែកខាងលើរបស់ web.phpជាមុនសិនដោយប្រើនូវ **use keyword**។

```
PostController.php web.php ×
routes > web.php
1 <?php
2
3 use App\Http\Controllers\PostController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('post/example', [PostController::class, 'example']);
7 Route::get('post/example1', [PostController::class, 'example2']);
```

ឧទាហរណ៍៖

- បង្កើតcontroller ដោយបញ្ជា php artisan make:controller PostController
- ចូលទៅកាន់controller class ដែលបានបង្កើតរួចសរសេរកូដដូចខាងក្រោម

```
PostController.php ×
app > Http > Controllers > PostController.php > PostController
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class PostController extends Controller
8 {
9     public function example()
10     {
11         return "this is first public method in controller.";
12     }
13
14     public function example2()
15     {
16         return "this is second public method in controller.";
17     }
18 }
```

- បន្ទាប់មកregister routeដូចខាងក្រោម៖

The image shows a step-by-step process of creating a Laravel controller and testing it via a web browser.

File Explorer: The left sidebar shows the project structure. `PostController.php` is highlighted in the `Controllers` directory, and `web.php` is highlighted in the `routes` directory.

PostController.php: The code defines a `PostController` class extending `Controller`. It contains two public methods: `example()` and `example2()`.

```

4 namespace App\Http\Controllers;
5 use Illuminate\Http\Request;
6
7 class PostController extends Controller
8 {
9     public function example()
10     {
11         return "this is first public method in controller.";
12     }
13
14     public function example2()
15     {
16         return "this is second public method in controller.";
17     }
18 }
    
```

web.php: The code defines two GET routes: `post/example` pointing to `PostController::example` and `post/example1` pointing to `PostController::example2`.

```

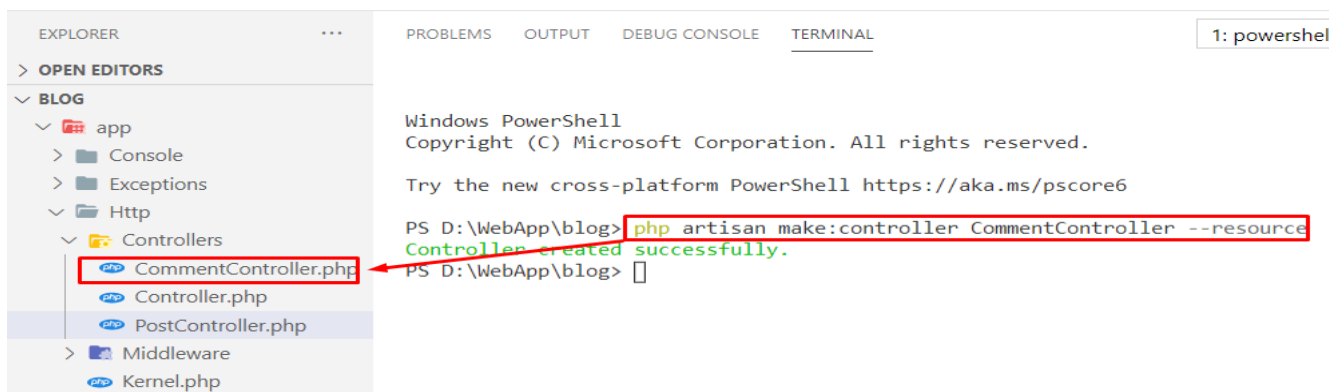
1 <?php
2
3 use App\Http\Controllers\PostController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::get('post/example', [PostController::class, 'example']);
7 Route::get('post/example1', [PostController::class, 'example2']);
    
```

Browser Window 1: The address bar shows `127.0.0.1:8000/post/example`. The page content is `this is first public method in controller.`

Browser Window 2: The address bar shows `127.0.0.1:8000/post/example1`. The page content is `this is second public method in controller.`

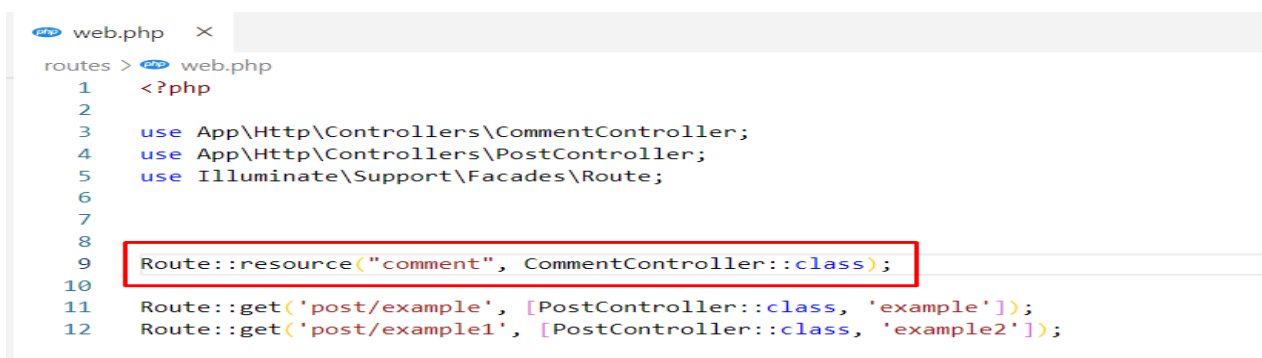
- Controller Resource

Laravel resource controllers វាផ្តល់នូវ CRUD(create, retrieve, updateនិងdelete) routes អោយ controller តែ single line of code។ resource controller ត្រូវបានប្រើដើម្បីបង្កើតcontroller ដែល handles the http requestsទាំងអស់។



- ការបង្កើតroute resource

ដើម្បីបង្កើត route resource គឺយើងត្រូវប្រើនូវ resource(\$name,\$controller) static method របស់ Route class។ដើម្បីរាយឈ្មោះ uri និង route name របស់resource route គឺប្រើបញ្ជា php artisan route:list ឬ php artisan r:l។



2.4. អំពី RequestនិងResponse

Request ជាព័ត៌មានផ្សេង ដែលកើតឡើងដោយអ្នកប្រើប្រាស់ ឬអ្នកស្នើសុំតាមរយៈ browserដែលវាត្រូវបានគ្រប់គ្រងដោយ Http Request objectរបស់កម្មវិធី។ដូចនេះគេអាចaccessនូវ current http requestបានតាមរយៈ dependency injectចាំបាច់ត្រូវវាយបញ្ចូលនូវ Illuminate\Http\Request class ចូលទៅក្នុងcontrollerជាមុនសិនឬក៏ប្រើប្រាស់នូវ request() helper method។Incoming request instance និងinjected ដោយស្វ័យប្រវត្តិតាមរយៈ service container។


```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class UserController extends Controller
{
    public function store(Request $request)
    {
        $name = $request->input('name');
        //
    }
}
```

```
web.php PostController.php X
app > Http > Controllers > PostController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 // use Illuminate\Http\Request;
6
7 class PostController extends Controller
8 {
9     public function example(\Illuminate\Http\Request $request)
10     {
11         return $request->id;
12     }
13
14     public function example2()
15     {
16         return "this is second public method in controller.";
17     }
18 }
```

```
app > Http > Controllers > CommentController.php > CommentController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class CommentController extends Controller
8 {
9
10     public function index()
11     {
12         return request()->id;
13     }
14
15     /**
16      * Show the form for creating a new resource.

```

Response

គ្រប់ routesនិង controllerគួរតែបញ្ជូនចំណេញតបត្រឡប់ទៅកាន់user's browserវិញ។ laravelបានផ្តល់វិធីសាស្ត្រជាច្រើនក្នុងការបញ្ជូនចំណេញតប(response)ត្រឡប់វិញមានដូចជា៖

- ✚ String និង array
- ✚ Redirect
- ✚ Json
- ✚ Views ជាដើម

EX). បោះតំលៃជាអក្សរនិងតំរៀប(string and array)

Web.php

```
<?php
```

```
use App\Http\Controllers\PostController;
```

```
use Illuminate\Support\Facades\Route;
```

```
Route::get('post/string', [PostController::class, 'string']);
```

```
Route::get('post/array', [PostController::class, 'array']);
```

PostController.php

```
<?php
```

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

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

```
class PostController extends Controller
```

```
{
```

```
    public function string()
```

```
    {
```

```
        return "this response is string.";
```

```
    }
```

```
    public function array()
```

```
    {
```

```
        return [1, 2, 3, 4, 5, 6, 8];
```

```
    }
```

```
}
```

មេរៀនទី៣ View និង Blade

1. អ្វីទៅជា View ?

View គឺជា representation layer

ដែលមានតួនាទីក្នុងបង្ហាញព័ត៌មានផ្សេងៗអោយអ្នកប្រើប្រាស់មើលឃើញនិងធ្វើអន្តរកម្មជាមួយវាបាន។ View ជា files ដែលមាន html markup, css និង JavaScript ។ គ្រប់ view files ទាំងអស់ត្រូវរក្សាទុកក្នុងថតមួយគឺ resources/views ។ View អាចត្រូវបានដាក់បញ្ចូលក្នុងថតរង(sub-directory)នៃថតឯកសារ resources/views ។ ការកំណត់ "." អាចត្រូវបានប្រើដើម្បីយោងនូវ nested view (Views may also be nested within subdirectories of the resources/views directory. "Dot" notation may be used to reference nested views) ។

1. អ្វីទៅជា Blade Template ?

ជា laravel engine ដែលអនុញ្ញាតិអោយយើងសរសេរ php code ជាមួយ html markup language ដោយផ្ទាល់បាននិងមានភាពងាយស្រួលហើយសម័ញ្ញ។ blade template engine វាផ្តល់រចនាសម្ព័ន្ធផ្ទាល់ខ្លួនរបស់វាដូចជាបញ្ហា conditional និង loops ជាដើម។ ដើម្បីបង្កើត blade template យើងគ្រាន់តែបង្កើត view file ហើយរក្សាទុកដែលមានទំរង់ *.blade.php ។

2. អ្វីទៅជា Form Handling ?

+ អំពី CSRF

CSRF មកពីពាក្យថា Cross Site Request Forgery វាជាការវាយប្រហារតាមវេបសាយ(web attack)ដែលបង្ខំអោយអ្នកប្រើប្រាស់ឱ្យធ្វើសំណើដោយអចេតនាទៅកម្មវិធីគេហទំព័រដែលពួកគេត្រូវបានផ្ទៀងផ្ទាត់ភាពត្រឹមត្រូវពីមុន(that forces a user to make unintended requests to a web application where they are previously authenticated) ។ រាល់ពេលដែលបង្កើតនូវ HTML form ជាមួយ application យើងត្រូវតែបញ្ចូលនូវ hidden CSRF token field ជាមួយ form ធ្វើដូច្នេះ CSRF protection middleware ធ្វើ validate នូវ request ដោយប្រើនូវ @csrf Blade directive ដើម្បីបង្កើតនូវ token field ។

+ អំពី Form Method

ដោយសារHTMLform មិនអាចបង្កើតសំណើរសុំ PUT, PATCH ឬ DELETE បានដូច្នេះអ្នកត្រូវបន្ថែម hidden fieldមួយដែលមានតំលៃស្មើនឹង(_method)ដើម្បីបង្ខំឱ្យHTTP verbsទាំងនេះដោយប្រើប្រាស់នូវ@method Blade អាចបង្កើតវា។

+ ការចាប់យកព័ត៌មានពីForm

ដើម្បីចាប់ព័ត៌មានពី form laravel បានផ្តល់វិធីសាស្ត្រចំនួនពីរដើម្បី access នូវuser data ជាមួយ laravel frameworkគឺ៖ តាមរយៈ request() global helper method និង injecting a request object(Illuminate\Http\Request)។ តារាងខាងក្រោមជា method សំខាន់ៗដែលគេប្រើសម្រាប់access user data៖

Methods	description
<code>\$request->all()</code>	សម្រាប់ចាប់យកតំលៃទាំងអស់ពីrequest ជាkey array
<code>\$request->input('field_name');</code>	បោះតំលៃរបស់អថេរ input(input variable)
<code>\$request->only(array \$keys);</code>	សម្រាប់ចាប់យកតំលៃតែផ្នែកណាមួយនៃrequestជា key array
<code>\$request->except(array \$keys);</code>	សម្រាប់យកតំលៃតែផ្នែកណាមួយនៃrequestលែងតែអថេរដែលមានexcept()ជា key array
<code>\$request->has(\$key)</code>	ពិនិត្យមើលថា តើrequestមានឈ្មោះនោះឬអត់
<code>\$request->filled(\$key)</code>	ពិនិត្យមើលថា តើrequestមានតំលៃឬអត់

+ ការUpload File

ដើម្បីធ្វើការជាមួយfile Laravel frameworkបានផ្តល់នូវapi ជាច្រើនដូចមានខាងក្រោម៖

- `$request->hasFile($key)`
 - ពិនិត្យមើលថាមានfile ឬអត់
- `$request->file($key)->isValid()`
 - ពិនិត្យមើលថាមានសុភលភាពឬអត់(validate)
- `$request->file($key)->extension()`
 - បោះតំលៃជាfile extension
- `$request->file($key)->path()`

- បោះតំលៃជាfile path
- \$request->file(\$key)->store(\$path,...)
 - ជាmethod សម្រាប់store file
- \$request->file(\$key)->storeAs(\$path,\$name,\$options)
 - ជាmethod សម្រាប់store file
- \$request->file(\$key)->getClientOriginalName()
 - បោះតំលៃជាឈ្មោះរបស់file
- \$request->file(\$key)->getClientOrinalExtension()
 - បោះតំលៃជាfile extension
- \$request->file(\$key)->getClientSize()
 - បោះតំលៃជាវិមាឌរបស់file

4. Validation

Laravel framework បានផ្តល់វិធីសាស្ត្រផ្សេងៗគ្នាដែលអាចអោយគេធ្វើការពិនិត្យនូវភាពត្រឹមត្រូវទិន្នន័យ (validate user's input)របស់ user's request មុននិងបញ្ជូនទៅកាន់ systemជាមួយ rules ជាច្រើនដើម្បីធ្វើការ validate នូវ user's request។base controller class ប្រើនូវ ValidatesRequests traitដែលបានផ្តល់នូវវិធីសាស្ត្រងាយស្រួលក្នុងការពិនិត្យភាពត្រឹមត្រូវនៃ HTTP requestជាមួយ power validation ruleផ្សេងៗ។

+ Rules មួយចំនួនដែលគេប្រើ

Available Validation Rules in Laravel		
Accepted	Active URL	After (Date)
Alpha	Alpha Dash	Alpha Numeric
Array	Before (Date)	Between
Boolean	Confirmed	Date
Date Format	Different	Digits
Digits Between	E-Mail	Exists (Database)
Image (File)	In	Integer
IP Address	JSON	Max

MIME Types(File)	Min	Not In
Numeric	Regular Expression	Required
Required If	Required Unless	Required With
Required With All	Required Without	Required Without All
Same	Size	String
Timezone	Unique (Database)	URL

+ ការប្រើ validate () លើ Request Object

ជាមួយ Request object វាបានផ្តល់ validate() method

មួយដែលផ្តល់ទំរង់កាត់និងផ្តល់ភាពងាយស្រួលក្នុងការ validate នូវ request ហើយវានិង redirect

ដោយស្វ័យប្រវត្តិនៅពេលដែល validate មិនបានជោគជ័យ។ រាល់ errors ទាំងអស់និងភ្ជាប់ជាមួយនិង view

តាមរយៈ Illuminate\View\Middleware\ShareErrorsFromSession

middleware ដែលរក្សាទុកក្នុងអថេរមួយគឺ៖ \$errors ។

រូបមន្ត៖ \$request->validate(\$rules,...\$params);

+ ការប្រើ validate() របស់ controller

យើងដឹងហើយថា base controller class បាន inherit នូវ ValidatesRequests trait ដូច្នេះហើយវាទទួលបាននូវ validate() method ដែរ។

រូបមន្ត៖ `$this->validate($request, array $rules, array $message=[]);`

+ Manually validation

ប្រសិនបើយើងមិនចង់ប្រើ validate() method របស់ Request object ឬ Controller ទេយើងអាចបង្កើត validator instance ដោយខ្លួនឯងដោយប្រើនូវ validator facade ស្ថិតនៅក្នុង Illuminate\Support\Facades\Validator namespace ។ ការធ្វើ manual validation វាមិនធ្វើការ redirect ដោយស្វ័យប្រវត្តិទេ។

រូបមន្ត៖ `$validator=Validator::make($request,array $rules,array $message=[]);`

+ អំពី Form Request

FormRequest គឺជា custom request classes ដែលអាចអោយយើងធ្វើការ validate និង authorize ក្រៅ controller ។ FormRequest ត្រូវបង្កើតដោយបញ្ជា៖

C:> php artisan make:request your_request_name

FormRequest class វាស្ថិតនៅក្នុងថត app/http/request ។ នៅក្នុង Formrequest class មាន methods សំខាន់ពីរគឺ៖

- authorize() បោះតម្លៃជា boolean
- rules() បោះតម្លៃជា array

មេរៀនទី៤ Database និង Eloquent

1. សេចក្តីផ្តើម

ឡាវ៉ាដែលធ្វើអន្តរកម្មជាមួយ database យ៉ាងសាមញ្ញបំផុតជាមួយនិង database backend ផ្សេងៗដោយប្រើនូវ Raw SQL, fluent query builder និង Eloquent ORM ។ ក្រោយមូលដ្ឋានទិន្នន័យផ្សេងៗគ្នាដោយប្រើទាំង SQL នៅអ្នកបង្កើតសំណួរដែលស្ទាត់ជំនាញនិងអរម៉ូនអ៊ែនជ្រួល។ បច្ចុប្បន្នឡាវ៉ាលក់ទ្រមូលដ្ឋានទិន្នន័យចំនួនបួនគឺ ៖

- MySQL 5.6+
- PostgreSQL 9.4+
- SQLite 3.8.8+
- SQL Server 2017

2. Configuration

ដើម្បីប្រើប្រាស់ database បានជាដំបូងត្រូវ Config database ជាមុនសិន។ ការ Config database ស្ថិតនៅ Config/database.php ដែលជាមានឈ្មោះថា .env file។ របៀប config database ៖

APP_NAME=Laravel

APP_ENV=local

APP_KEY=base64:BpTlylp6+FtlqKKJ+ScUyBcp+pHIM3q7tNFizSvxvEw=

APP_DEBUG=true

APP_URL=http://localhost

LOG_CHANNEL=stack

LOG_DEPRECATED_CHANNEL=null

LOG_LEVEL=debug

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=laravel

DB_USERNAME=root

DB_PASSWORD=



BROADCAST_DRIVER=log

CACHE_DRIVER=file

FILESYSTEM_DRIVER=local

QUEUE_CONNECTION=sync

SESSION_DRIVER=file

SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1

REDIS_PASSWORD=null

REDIS_PORT=6379

MAIL_MAILER=smtp

MAIL_HOST=mailhog

MAIL_PORT=1025

MAIL_USERNAME=null

MAIL_PASSWORD=null

MAIL_ENCRYPTION=null

MAIL_FROM_ADDRESS=null

MAIL_FROM_NAME="\${APP_NAME}"

AWS_ACCESS_KEY_ID=

AWS_SECRET_ACCESS_KEY=

AWS_DEFAULT_REGION=us-east-1

AWS_BUCKET=

AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=

PUSHER_APP_KEY=

PUSHER_APP_SECRET=

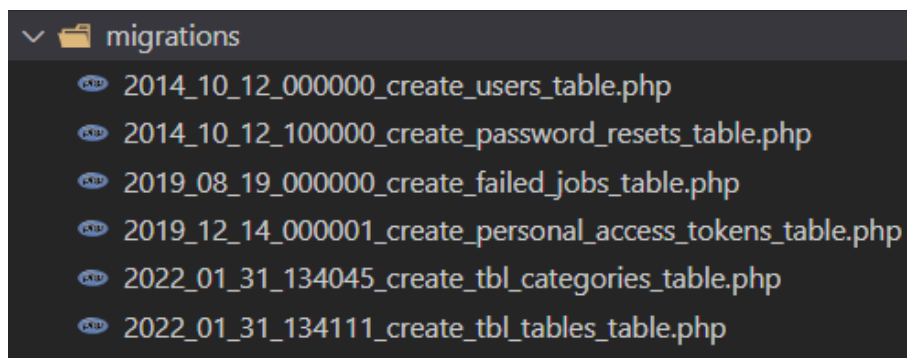
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="\${PUSHER_APP_KEY}"

MIX_PUSHER_APP_CLUSTER="\${PUSHER_APP_CLUSTER}"

2.1. Migration

Migration ជាវិធីសាស្ត្រមួយដែលអោយយើងបង្កើត database's table ដោយប្រើប្រាស់នូវ php script ហើយវាក៏អនុញ្ញាតិអោយយើងធ្វើការកែប្រែឬចែករំលែក(modify and share)នូវ database schema ផងបានដែរ។ គ្រប់ migration files ទាំងអស់ស្ថិតនៅក្នុងថត databases/migrations ។



2.2.1. ការបង្កើត Migration

មុននឹងអាច modify ឬ share database បានចាំបាច់ត្រូវបង្កើត migration file ជាមុនសិន។ គេមានបញ្ជាមួយចំនួន ដើម្បីបង្កើត migrations ដូចជា៖

- php artisan make:migration your_migration_name
 - សម្រាប់បង្កើត migration file
- php artisan make:migration your_migration_name --create=table_name
 - សម្រាប់បង្កើត migration file ដែលឈ្មោះតារាងអាស្រ័យលើ --create flag
- php artisan make:migration your_migration_name --table=TableName
 - សម្រាប់បង្កើត migration file ដើម្បី modify table's structure ។

Example:

```

MINGW64:/d/DOC-IT/Laravel Project/AssignmentTeam
code .
ASUS@DESKTOP-TPDPLHE MINGW64 /d/DOC-IT/Laravel Project/AssignmentTeam
$ code .
ASUS@DESKTOP-TPDPLHE MINGW64 /d/DOC-IT/Laravel Project/AssignmentTeam
$ php artisan make:migration create_todolist_table
Created Migration: 2022_02_13_075855_create_todolist_table

```

2.3 ការបង្កើតតារាង (table)

យើងប្រើ table() method របស់ Schema façade ដើម្បីធ្វើការកែប្រែនូវរចនាសម្ព័ន្ធរបស់តារាងដែលមានស្រាប់។ ដូច create() method ដែរ ត្រូវការ argument ចំនួនពីរគឺ៖

- ឈ្មោះរបស់តារាង
- និងទី២គឺ function ដែលវាទទួលយក Blueprint object ដែលប្រើសម្រាប់កំណត់លក្ខណៈអោយតារាង។

Example:

```
public function up()
{
    Schema::create('tbl_categories', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('image');
        $table->enum('status', array('active', 'inactive'));
        $table->timestamps();
    });
}
```

2.4. Constrains

- \$table->primary(\$column);
 - បង្កើត primary column
- \$table->foreign(\$column)->references(\$column)->on(\$table)->onDelete('.....')->onUpdate('....');
 - បង្កើតទំនាក់ទំនង
- \$table->string(\$column)->nullable();
 - បង្កើត column ដែលមានតំលៃ null
- \$table->string(\$column)->Unique();
 - បង្កើត column ដែលមិនមានតំលៃស្មើ
- \$table->string(\$column)->default("Male");
 - បង្កើត column ដែលមានតំលៃស្រាប់
- \$table->index(\$column)

Example:


```

public function up()
{
    Schema::create('books', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->string('page');
        $table->longText('description');
        $table->unsignedBigInteger('author_id');
        $table->foreign('author_id')->references('id')->on('author');
        $table->timestamps();
    });
}

```

2.5 ការដំឡើង migration

ដើម្បី update នូវ database schema ត្រូវតែដំឡើងការ migration ជាមុនសិន ខាងក្រោមជាបញ្ជាមួយចំនួន សម្រាប់ដំឡើងការ migration។

- php artisan migrate
 - Use to run all migrations file by running up() method
- php artisan migrate --seed
 - Use to run all migrations file by running up() method with seed data
- php artisan migrate:refresh
 - Drop all table and run every migrations again
- php artisan migrate:refresh

2.6 Running Raw sql query

ពេលដែលយើងបាន configured database connection រួចហើយ អាចអោយយើង run queries ដោយប្រើ DB facade. DB facade វាបានផ្តល់នូវ methods មួយចំនួនដូចជា:

- select
- update
- insert
- Delete
- នឹង statement.

3. Query builder

Laravel's database query builder provides a convenient, fluent interface to creating and running database queries. It can be used to perform most database operations in your application and works on all supported database systems. The Laravel query builder uses PDO parameter binding to protect your application against SQL injection attacks.

3.1 retrieves all row

Syntax:

- `DB::table($table)->get();`
 - ទាញយកគ្រប់ rows និង columns
- `DB::table($table)->select($columns)->get()`
 - ទាញយកគ្រប់ rows និង columns ខ្លះៗ។

Example:

```
class PageController extends Controller
{
    public function webpage(){
        return view('web.web');
    }
    public function db(){
        $id = 2;
        $posts = DB::table('toolist')
        // return view('welcome', compact('posts'));
        // ->where('product_id', $id)->delete();
        // ->where('product_id', $id)->update(['product_name'=>'Tholsotheara', 'Qty'=>'
        ->insert(['name'=>'Myura', 'decription'=>'20000', 'complete'=>'1']);
        // >get();
        dd($posts);
    }
}
```

3.3 Aggregate

The query builder also provides a variety of aggregate methods such as count, max, min, avg, and sum.

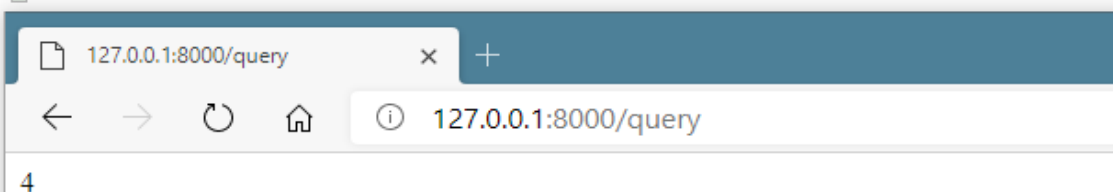
- `$users = DB::table(string $table)->count();`
- `$price = DB::table(string $table)->max(string $column);`
- `$users = DB::table(string $table)->sum(string $column);`
- `$price = DB::table(string $table)->min(string $column);`
- `$price = DB::table(string $table)->avg(string $column);`

routes > web.php > #Function#a5669980

```

1  <?php
2
3  use App\Models\Author;
4  use App\Models\Book;
5  use App\Models\Student;
6  use Illuminate\Support\Facades\DB;
7  use Illuminate\Support\Facades\Route;
8
9  Route::get('query', function () {
10     $counts = DB::table('students')->count();
11     return $counts;
12 });
13
14
15
16
17

```



3.4.Insert Delect Data

- DB::table(string \$table)->delete();
 - លុបទិន្នន័យទាំងអស់ពីតារាង
- DB::table(string \$table)->where(\$condition)->delete();
 - លុបទិន្នន័យចេញពីតារាងតាមលក្ខខណ្ឌ
- DB::table(string \$table)->truncate()
 - លុបទិន្នន័យចេញពីតារាងតាមលក្ខខណ្ឌហើយreset auto-increment
- DB::table(string \$table)->insert(array \$keys);
 - បញ្ចូលម្តងមួយrow
- DB::table(string \$table)->insert(array(array \$keys))
 - បញ្ចូលច្រើន rows ក្នុងពេលតែមួយ
- DB::table(string \$table)->insertGetId(array \$keys)
 - បញ្ចូលម្តងមួយ row ហើយបោះតំលៃ id(auto-increment)

Example:

```

class PageController extends Controller
{
    public function webpage(){
        return view('web.web');
    }
    public function db(){
        $id = 2;
        $posts = DB::table('toolist')
        // return view('welcome', compact('posts'));
        // ->where('product_id', $id)->delete();
        // ->where('product_id', $id)->update(['product_name'=>'Tholsotheara',
        ->insert(['name'=>'Myura', 'decription'=>'20000', 'complete'=>'1']));
        // >get();
        dd($posts);
    }
}

```

4. Eloquent

Eloquent ORM(Object Relational Mapping) ដែលមានតួនាទីក្នុងការធ្វើអន្តរកម្មរវាង application ជាមួយជាតាបេស(insert, select, update និង delete ទិន្នន័យ)។ Database table ដែលត្រូវគ្នាជាមួយ "Model" ដែលប្រើដើម្បីធ្វើអន្តរកម្មជានិងតារាងនោះ(Each database table has a corresponding "Model" which is used to interact with that table)។ Model អោយយើងធ្វើការជាមួយទិន្នន័យជាមួយនិង database table(Models allow you to query for data in your tables)និងបញ្ចូលទិន្នន័យផងដែរ(as well as insert new records into the table)។

4.1. Model

ជាមួយឡាវ៉ាល Model គឺជាclassមួយដែលតំណាងឱ្យរចនាសម្ព័ន្ធ logical និងទំនាក់ទំនងនៃតារាងទិន្នន័យមូលដ្ឋាន។ នៅក្នុងLaravel តារាងទិន្នន័យនីមួយៗ(each database table)ដែលដែលត្រូវគ្នាជាមួយ "Model" នីមួយៗដែលអនុញ្ញាតឱ្យយើងធ្វើអន្តរកម្មជាមួយតារាង(table)នោះ។

Models ទាំងអស់ត្រូវស្ថិតនៅក្នុងថត App\Models ដើម្បីបង្កើត Modelគេមានបញ្ហាមួយចំនួនដូចខាងក្រោម៖

- php artisan make:model ModelName
 - បង្កើត model
- php artisan make:model ModelName --migration
 - បង្កើត model និង migration file
- php artisan make:model ModelName -m
 - បង្កើត model និង migration file
- php artisan make:model ModelName -mc
 - បង្កើត model migration file និង controller

Example:

```
ASUS@DESKTOP-TPDPLHE MINGW64 /d/DOC-IT/Laravel Project/Toolist (master)
$ php artisan make:model todolist
Model created successfully.

ASUS@DESKTOP-TPDPLHE MINGW64 /d/DOC-IT/Laravel Project/Toolist (master)
$ !
```

4.2.Eloquent Model conventions

- Table Names

- ជាទូទៅ Model និងធ្វើទំនាក់ទំនងជាមួយតារាងតាមរយៈ plural name of the class
- តែគេក៏អាចប្តូរតារាងបានដែរគឺ៖

protected \$table="your_table_name";

4.3. Retrieve Model

- ទាញយក models ទាំងអស់
 - Model::all();
 - Model::get();
 - Model::where(\$condition)->get();
- ទាញយកតែមួយ model
 - Model::first()
 - Model::firstOrFail()
 - Model::find(\$id)
 - Model::findOrFail(\$id)
- ការប្រើប្រាស់ aggregate
 - Model::count();
 - បោះតំលៃចំនួន models ដែលមាន
 - Model::max(string \$column)
 - បោះតំលៃធំបំផុតរបស់ model អាស្រ័យលើ column
 - Model::min(string \$column)
 - បោះតំលៃតូចបំផុតរបស់ model អាស្រ័យលើ column
 - Model::sum(string \$column)
 - បោះតំលៃផលបូករបស់ model អាស្រ័យលើ column
 - Model::avg(string \$column)

- បោះពុម្ពផ្សាយកម្រិតមធ្យមលើ column

Example:

```
web.php x
routes > web.php > #Function#a5669980
1 <?php
2
3 use App\Models\Author;
4 use App\Models\Book;
5 use App\Models\Student;
6 use Illuminate\Support\Facades\DB;
7 use Illuminate\Support\Facades\Route;
8
9 Route::get('query', function () {
10 //=====retrieve all models=====
11 $alls = Student::all(); //retrieve all models
12 $gets = Student::get(); //retrieve all models
13 $where = Student::where('id', 3);
14 //=====retrieve single model=====
15 $found1 = Student::find(3); //retrieve one model by primary key
16 $found2 = Student::findOrFail(3); //retrieve one model by primary key with error
17
18 $first = Student::first();
19 $firstOrFail = Student::findOrFail();
20 //=====
21
22 });
--
```

4.4 Create and update model

- ដើម្បីបង្កើត record ថ្មីគឺគេបង្កើត new instance model
- បន្ទាប់មក ផ្តល់តំលៃអោយ properties នីមួយៗរបស់ model
- បន្ទាប់មកហៅ save() method។

```
$author=authors::findOrFail($req->author_id);
$books = new books();
$books->title = $req->input('title');
$books->page = $req->input('page');
$books->description = $req->input('description');
$booksresult = $author->books()->save($books);
return redirect()->route("library");
```

- Update model

- ដើម្បីបង្កើតកែប្រែទិន្នន័យគឺដំបូងត្រូវ ស្វែងរកទិន្នន័យជាមុនសិន
- បន្ទាប់មកផ្តល់តំលៃថ្មីអោយ properties នីមួយៗ
- ចុងក្រោយហៅ save() method។

Example:

```
public function update(Request $req, $id)
{
    $data = books::where('id',$id)->update([
        'title' => $req->title,
        'page' => $req->page,
        'authors_id'=>$req->author_id,
        'description'=>$req->description,
    ]);
    if($data){
        return redirect("/library");
    }
}
```

4.5 Delete Model

- លុប model ដោយ delete() method តាមរយៈ model instance
 - \$model->delete();
- លុប model តាមរយៈ model primary key
 - Model::destroy(\$id)
 - Model::destroy(...\$id);
 - Model::destroy([...\$id]);

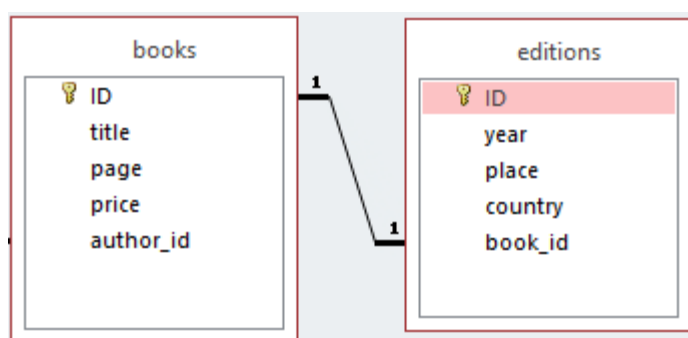
```
web.php ×
routes > web.php > ...
1  <?php
2
3  use App\Models\Author;
4  use App\Models\Book;
5  use App\Models\Student;
6  use Illuminate\Support\Facades\Route;
7
8  Route::get('query', function () {
9
10     $student = Student::find(1);
11     $student->delete();
12 });
```

5. Relation

Database table តែងតែមានទំនាក់ទំនងទៅវិញទៅមក។ ឧទាហរណ៍ blog post មួយមានទំនាក់ទំនងជាមួយ comments ជាច្រើន។ eloquent ធ្វើការគ្រប់គ្រងនិងតំណើរការជាមួយទំនាក់ទំនងដោយស្រួលហើយទ្រទ្រង់នូវប្រភេទទំនាក់ទំនងប្រភេទផ្សេងគ្នាដូចជា៖

- One to one
- One to many
- Many to many ជាដើម។

5.1 One to One



One to one relationship មានន័យថា model មួយមានទំនាក់ទំនងជាមួយ model មួយទៀត។ ដើម្បីបង្កើតទំនាក់ទំនងត្រូវ៖

- ត្រូវកំណត់ថាមួយណាជា principle model និងមួយណាជា dependent model
- បន្ទាប់មកត្រូវបង្កើត method មួយក្នុង principle model

នៅពេលដែលបង្កើតបង្កើត method, method ត្រូវតែជាប្រភេទ return type ជាមួយនិងប្រភេទដូចខាងក្រោម៖

- return \$this->hasOne(Related Model);
- return \$this->hasOne(Related Model,foreign key);
- return \$this->hasOne(Related Model,foreign key,local key)

- Revers Model Relation

យើងអាច access នូវ parent model ពី related model បានដោយយើងបង្កើត method មួយនៅក្នុង related model ជាមួយ belongTo() method។

5.2 One to many

One to Many relationship មានន័យថា model មួយមានទំនាក់ទំនងជាមួយ model ច្រើនទៀត។ ដើម្បីបង្កើតទំនាក់ទំនងត្រូវ៖

- ត្រូវកំណត់ថាមួយណាជា principle model(parent)និងមួយណាជា dependent (related)t model។
- បន្ទាប់មកត្រូវបង្កើត method មួយក្នុង principle model។

នៅពេលដែលបង្កើតបង្កើត method, method ត្រូវតែជាប្រភេទ return typeជាមួយនិងប្រភេទដូចខាងក្រោម៖

- return \$this->hasMany(Related Model);
- return \$this->hasMany(Related Model,foreign key);
- return \$this->hasMany(Related Model,foreign key,local key)

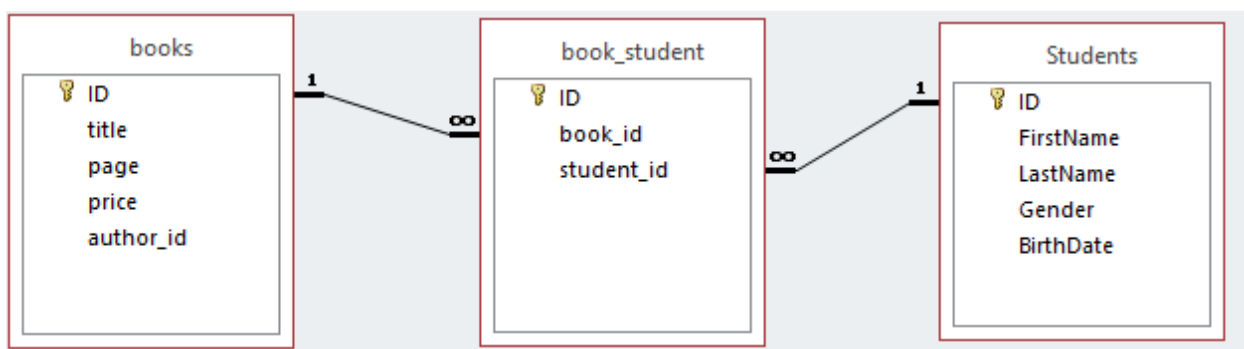
5.3 Many to Many

Many to Many relationshipមានន័យថា model មួយមានទំនាក់ទំនងជាមួយ modelច្រើនហើយត្រឡប់មកវិញហើយវាមានលក្ខណៈស្មុគស្មាញជាង hasOne()និងhasMany() method។ដូចនេះដើម្បីបង្កើតទំនាក់ទំនងត្រូវ៖

- ត្រូវបង្កើត intermediate tableមួយ(pivot table)ដោយ intermediate table មានកំណត់នូវ foreign key របស់ model(តារាង)ទាំងពីរ។
- បន្ទាប់មកត្រូវបង្កើត method មួយក្នុង model មួយណាដែលបង្កើតទំនាក់ទំនងដោយប្រើmethod belongToMany();

នៅពេលដែលបង្កើតបង្កើត method, method ត្រូវតែជាប្រភេទ return typeជាមួយនិងប្រភេទដូចខាងក្រោម៖

- return \$this->belongsToMany(Related Model);
- return \$this->belongsToMany(Related Model,\$pivot_table);
- return \$this->belongsToMany(Related Model,\$pivot_table,\$foreignPivottable,\$relatedPivotTable);



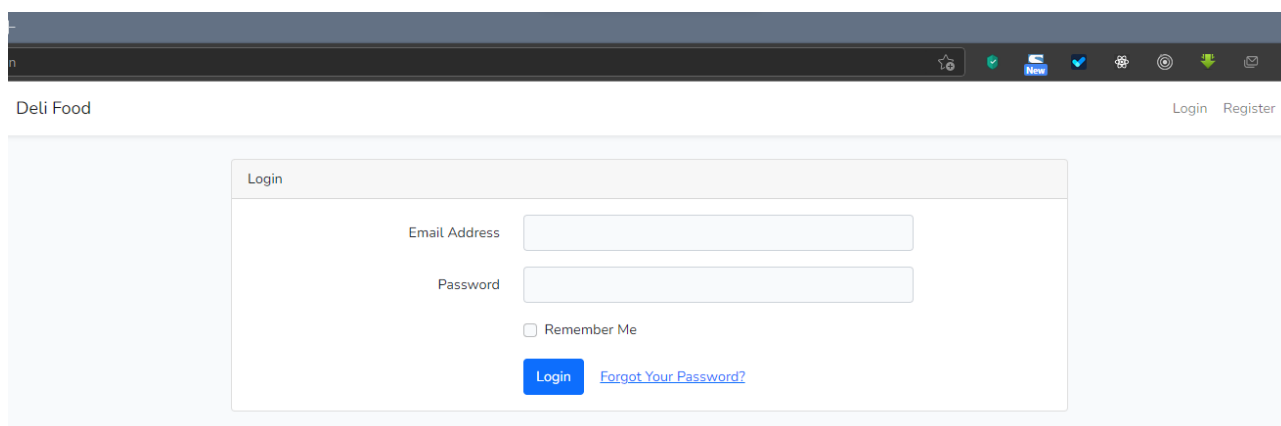
II. Restaurant Management System

2.1. Source Code

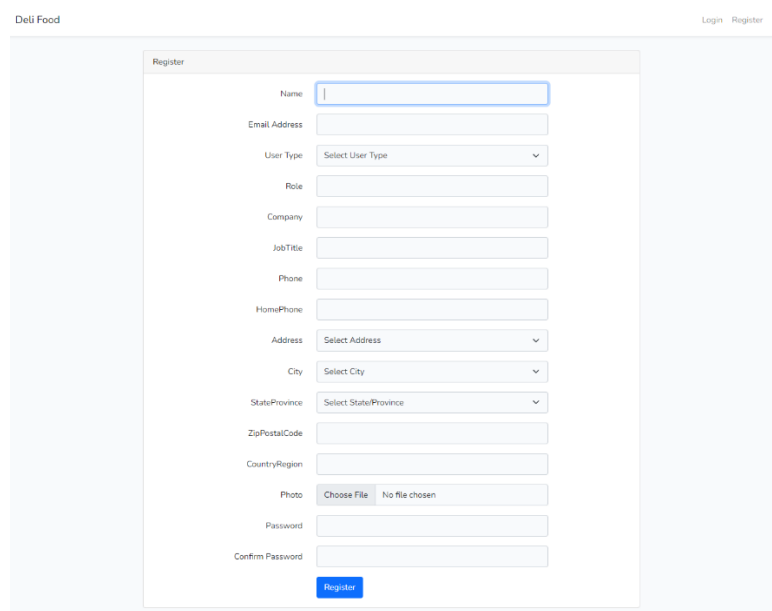
Please visit github: [LornSovannra/Restaurant-Management-System: Restaurant Management System \(github.com\)](https://github.com/LornSovannra/Restaurant-Management-System)

2.2. User Interface

2.2.1. Login Page



2.2.2. Register Page



2.2.3. Forgot Password Page

Deli Food

Login Register

Reset Password

Email Address

Send Password Reset Link

2.2.4. Reset Password Page

Deli Food

Login Register

Reset Password

Email Address

Password

Confirm Password

Reset Password

2.2.5. Dashboard Page

Dashboard
Order Detail
Order
Item
Note
Category
Employee

Welcome back, Lom Sovannal

Order Detail

4

Order

4

Item

7

Category

5

Employee

2

List of Items

Name: Angkor Puro
Price: \$25
Category: Soft Drink
Status: In-Stock

Name: Vital
Price: \$2
Category: Soft Drink
Status: Normal

Name: Dasani
Price: \$25
Category: Soft Drink
Status: Normal

Name: Coca Cola
Price: \$8
Category: Beverage
Status: In-Stock

2.2.6. Order Detail Page

Dashboard
Order Detail
Order
Item
Note
Category
Employee

Welcome back, Lom Sovannal

Create New Order Detail

Search

ID	Order ID	Item Name	Qty Order	Actions
1	1	Angkor Puro	5	
2	1	Coca Cola	2	
3	1	Fanta	3	
5	2	Purita	4	

Showing 1 to 4 of 4 entries

Previous 1 Next

2.2.6.1. Create Modal

Order Detail
×

Create

Order ID

Select Order ID

Item ID

Select Item ID

Qty Order

Close

Create

2.2.6.2. View Modal

Order Detail
×

View

ID

1

Order ID

1

Item ID

1

Qty Order

5

Close

2.2.6.3. Update Modal

Order Detail
×

Update

Order ID

1

Item ID

Angkor Puro

Qty Order


5

Close

Update

2.2.6.4. Delete Modal

Delete Order Detail
×



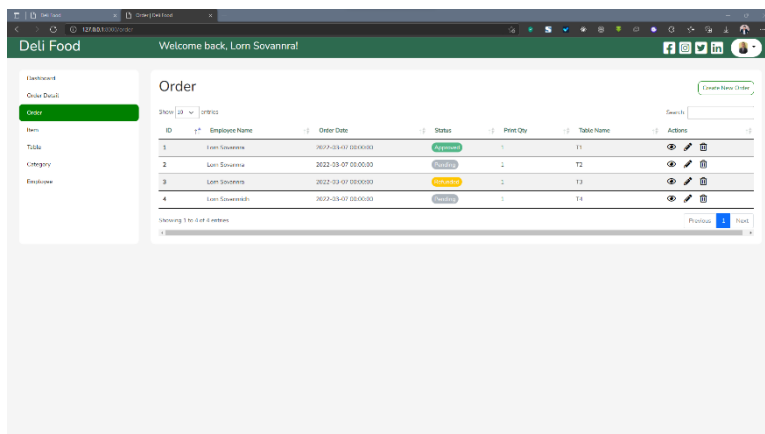
Are you sure?

You won't be able to revert this!

Yes, delete it!

Cancel

2.2.7. Order Page



2.2.7.1. Create Modal

Order
×

Create

Employee ID

Order Date

Status

Print Qty

Table ID

mm/dd/yyyy

Select Status

Select Item ID

Close

Create

2.2.7.2. View Modal

Order
×

View

Order ID

Employee ID

Order Date

Status

Print Qty

Table ID

1

1

2022-03-07 00:00:00

Approved

1

1

Close

2.2.7.3. Update Modal


Order ×

Update


Employee ID

1

Order Date

mm/dd/yyyy 


Status

Approved 

Print Qty

1






Table ID

T1 

Close

Update

2.2.8. Item Page

Deli Food Welcome back, Lorn Sovannral     

Dashboard

Order Detail

Order

Item




































Item

Category

Employee

Item

Create New Item

ID	Item Name	Description	Category Name	Unit Price	Status	Item Image	Actions
1	Angkor Pils		Soft Drink	\$0.35			  
2	Viet		Soft Drink	\$0.7			  
3	Eleven		Soft Drink	\$0.25			  
4	Guin Gale		Beerage	\$0.8			  
5	Fanta		Beerage	\$0.5			  
6	Chicken Fry		Fast Food	\$1.5			  
7	Springroll		Noodle	\$1.5			  

Showing 1 to 7 of 7 entries

Previous 1 2 Next

2.2.8.1. Create Modal


Item ×

Create

Item Name


Description

Category ID

Select Category ID 

Unit Price

Status

Select Status 

Item Image

Choose File

No file chosen


Close

Create

2.2.8.2. View Modal

Item

View



Item ID	1
Item Name	Angkor Puro
Description	
Category ID	1
Unit Price	0.25
Status	Hot

Close

2.2.8.3. Update Modal

Item

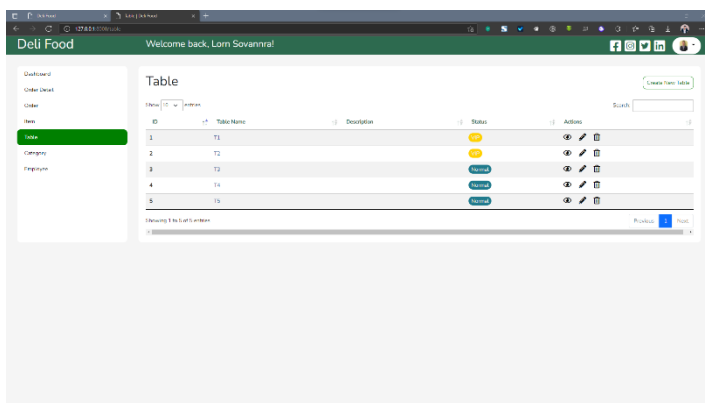
Update

Item Name	Angkor Puro
Description	
Category ID	Soft Drink
Unit Price	0.25
Status	Hot
Item Image	<div>Choose File</div> <div>No file chosen</div>

Close

Update

2.2.9. Table Page



2.2.9.1. Create Modal

Table

×

Create

Table Name

Description

Status

Select Status

▼

Close

Create

2.2.9.2. View Modal

Table

×

View

Table ID

1

Table Name

T1

Description

Status

VIP

Close

2.2.9.3. Update Modal

Table

×

Update

Table Name

T1

Description

Status

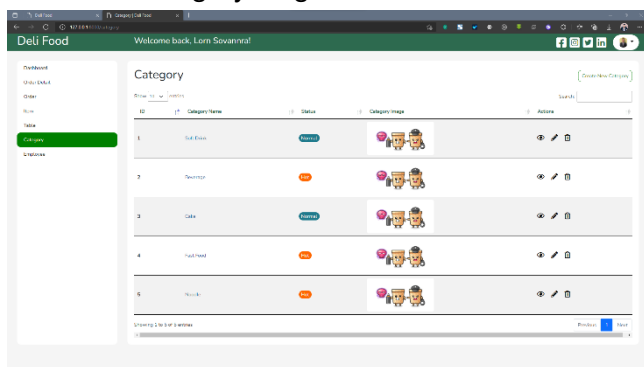
VIP

▼

Close

Update

2.2.10. Category Page



2.2.10.1. Create Modal

Category

Create

Category Name

Status

Select Status

Category Image

Choose File

No file chosen

Close

Create

2.2.10.2. View Modal

Category

View

ID

1

Category Name

Soft Drink

Status

Normal

Close

2.2.10.3. Update Modal

Category

Update

Category Name

Soft Drink

Status

Normal

Category Image

Choose File

No file chosen

Close

Update

2.2.11. Employee Page

Dashboard

Order

Item

Table

Category

Employee

Welcome back, Lern Savannrat

Employee

ID

Name

Email

User Type

Role

Company

Job Title

Phone

Home Phone

Address

City

State/Province

Zip/Postal

1

Lern Savannrat

lervannrat@gmail.com

Admin

CEO

Food

Software Developer

123456789

0123456789

123456789

123456789

123456789

123456789

123456789

123456789

2

Lern Savannrat

lervannrat@gmail.com

Admin

CEO

Food

Software Developer

123456789

0123456789

123456789

123456789

123456789

123456789

123456789

123456789

2.2.11.1. Create Modal

Employee

Create

Name

Email Address

User Type

Select User Type

Role

Company

Job Title

Phone

Home Phone

Address

Select Address

City

Select City

State/Province

Select State/Province

Zip/Postal Code

Country/Region

Photo

Choose File

No file chosen

Password

Confirm Password


Close

Create

2.2.11.2. View Modal

Employee
×

View



Employee ID	1
Name	Lorn Sovannra
Email Address	sovanra.lorn@gmail.com
User Type	Admin
Role	CTO
Company	Frodzy
Job Title	Software Developer
Phone	099 961 656
Home Phone	078 961 656
Address	Siem Reap
City	Siem Reap
State/Province	Siem Reap
Zip/Postal Code	17259
Country/Region	Cambodia

Close

2.2.11.3. Update Modal

Employee ×

Update

Name

Lorn Sovannra

Email Address

sovannra.lorn@gmail.com

User Type

Admin

Role

CTO

Company

Frodzy

JobTitle

Software Developer

Phone

099 961 656

HomePhone

078 961 656

Address

Siem Reap

City

Siem Reap

StateProvince

Siem Reap

ZipPostalCode

17259

CountryRegion

Cambodia

Photo

Choose File

No file chosen

Close

Update