Laravel

Laravel è un framework usato per il back-end realizzato in PHP. Grazie a questo framework possiamo semplificare molte azioni relative alla gestione del progetto e dunque aumentare la

* Riusabilità
* Leggibilità
* Manutenibilità
* Scalabilità

Per usare Laravel bisognerà scaricare composer in quanto ci permette, da prompt, di effettuare operazioni di gestione delle librerie relative al progetto.

Laravel è un framework MVC basato quindi sull’architettura Model View Controller.

* il model fornisce i metodi per accedere ai dati utili all'applicazione;
* il view visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
* il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

Una volta scaricato è possibile usare composer per creare il progetto. Successivamente si produce una directory con tanti file all’interno, relativi al progetto.

Tra questi file è presente “.env” contenente le variabili utilizzate da Laravel per operazioni relative ai DB, Debugging, Dominio, Mail ecc…

Quando si vuole effettuare qualche modifica all’interno di .env bisognerà farla in .env.example per non avere problemi dopo la sincronizzazione con gli altri programmatori. Sarà possibile, prendere eventuali variabili dal file .env.example successivamente.

php artisan [azione]: Famiglia di comandi per effettuare operazioni in grado di semplificare alcune attività di costruzione della web app. (Es: php artisan serve – Avvia la web app sul server impostato in .env) Molti altri comandi utili si trovano in questa lista:

make:channel Create a new channel class

make:command Create a new Artisan command

make:controller Create a new controller class

make:event Create a new event class

make:exception Create a new custom exception class

make:factory Create a new model factory

make:job Create a new job class

make:listener Create a new event listener class

make:mail Create a new email class

make:middleware Create a new middleware class

make:migration Create a new migration file

make:model Create a new Eloquent model class

make:notification Create a new notification class

make:observer Create a new observer class

make:policy Create a new policy class

make:provider Create a new service provider class

make:request Create a new form request class

make:resource Create a new resource

make:rule Create a new validation rule

make:seeder Create a new seeder class

make:test Create a new test class

Routes: Indica la cartella che gestisce i “percorsi” della web app. Ad esempio: /home. In routes, sono presenti già dei file in .php utili ai percorsi. Tra cui web.php che conterrà, in poche parole, tutte le “pagine” web dell’applicazione.

Lo scopo di routes, tuttavia, non è di identificare i percorsi ma anche di effettuare delle operazioni quando un utente naviga in quel percorso.

La sintassi è:

Route::get(‘/percorso’, function() {

//operazioni (Es. return view(‘pagina\_iniziale’))

}); //è possibile usare anche post anziché get, in base alle richieste che vogliamo gestire.

Oppure potremmo sostituire function() con il nome di un controller che fa determinate azioni specificate nel controllerstesso. (es. Route::get(‘/’, ‘NomeController@metodo’)

Curly Braces: è uno strumento che Laravel mette a disposizione quando si programmano le view. Serve ad usare codice PHP nello stesso file html senza aprire o chiudere ogni volta le parentesi <?php e ?>. Le doppie parentesi equivalgono ad un “echo” mentre la chiocciola serve per scrivere loop o condizioni (Es. @if(//condizione)).

Views: Cartella dove sono presenti i file .blade contenenti la struttura della pagina.

Controllers: Cartella dove sono presenti i “controller” oggetti che estendono la classe “Controller” e che definiscono delle funzioni richiamabili dalla view per processare dati.

Models: Cartella dove sono presenti tutti i “model” ovvero elementi in grado di manipolare e accedere ai dati. (Esempio “Studente” che estende Model.)

Migrations: Cartella dove sono presenti i file di migrazione, ovvero classi che interagiscono con il database per manipolarne i dati tramite operazioni di CRUD (Create, Read, Update, Delete).

PS: Per utilizzare MySQL con comodità è necessario l’uso di *Navicat.*

Php artisan make:migration nometabella: comando per creare una tabella del database. Una volta creata la tabella bisognerà necessariamente entrare nel file (in migrations) e andare a manipolare la funzione up() nel quale specificheremo lo “Schema::create” per la creazione della tabella con i vari attributi.

Variabili contenenti il risultato di query: Per effettuare query esistono vari modi. Il più semplice è usare SQL. Es: $posts=DB::select('select \* from users join posts on users.id = posts.user\_id where users.id = 1'). E’ possibile anche usare Eloquent, Lazy Eager, Eager Loading o Query Builder per fare la stessa azione. Bisogna tenere conto che per effettuare le query dobbiamo inserire il riferimento alla tabella nel Model in questione e inserire le query nei metodi del Controller che far riferimento al Modello. Inoltre ogni query viene assegnata ad una variabile la quale può essere manipolata tramite curly braces.

Altre info:

https://www.html.it/pag/55315/eloquent-lorm-di-laravel/

Strutturare il layout

Per sfruttare al meglio laravel, si utilizzano dei tag che permettono di risparmiare tempo nella creazione diverse componenti strutturali del sito. Solitamente si usa creare una pagina master.blade.php che contiene la struttura html. Le altre pagine possono “estendere” la pagina master e tramite delle “direttive” manipolare la pagina master.

Esempio:

Nella pagina master inseriamo <title> @yield(‘title’) </title>

Nelle pagine che estendono quella master possiamo scrivere la direttiva @section(‘title’, ‘titolo’)

Le pagine che estendono quella master non faranno altro che riprodurre la pagina master con la differenza che ogni pagina può modificare la sezione del titolo, ad esempio o cambiare altri contenuti della stessa pagina.

Solitamente in questi casi viene usati @yield e @section.

@yield(‘content’): descrive un’area manipolabile nel master blade

@section(‘content’): definisce il contenuto che andrà inserito nello yield (nella pagina che estende il master blade)

* Altre info: <https://stackoverflow.com/questions/41916127/whats-the-difference-between-laravels-yield-and-include>
* <https://laravel.com/docs/5.0/templates>

INSTALLAZIONE (Laravel + Laravel Homestead):

**Laravel Homestead** è una "scatola" ufficiale preconfezionata di Vagrant che ti fornisce un meraviglioso ambiente di sviluppo senza richiedere l'installazione di PHP, HHVM, un server web e qualsiasi altro software server sulla tua macchina locale. Non dovrai più preoccuparti di rovinare il tuo sistema operativo! Le scatole Vagrant sono completamente usa e getta. Se qualcosa va storto, puoi distruggere e ricreare la scatola in pochi minuti.

1. Scaricaricare “Composer” tramite set-up (Consigliato) anziché tramite cmd.
2. Creare il progetto laravel su una cartella in un path usando il comando “create project…”
3. Installare VirtualBox in modo da avere una Macchina Virtuale utilizzata per impostare il progetto in modalità Client/Server
4. Installare Vagrant che ci permette di interagire con la Macchina Virtuale.
5. Eseguire da CMD:

vagrant box add laravel/homestead

(Impiega 15-30 min)