MODEL VIEW CONTROLLER PATTERN

ARCHITETTURA A PIU' LIVELLI

Three-tier architecture ha i seguenti tre livelli:

1) Livello di presentazione

Mostra le informazioni relative a servizi come merce online, acquisti, e i contenuti del carrello della spesa.

2) Livello applicazione (business logic)

Viene tirato fuori dal livello di presentazione e controlla la funzionalità eseguendo elaborazioni dettagliate.

3) Livello dati

Le informazioni vengono salvate e lette dal server database.

ARCHITETTURA A PIU' LIVELLI

Three-tier architecture ha i seguenti tre livelli:

1) Livello di presentazione

In applicazione web based, il front-end è il contenuto (statico o dinamico) visualizzato dal browser.

2) Livello applicazione (business logic)

Processo di contenuti dinamici e generazione del livello di application server, per esempio PHP.

3) Livello dati

Un back-end database che gestisce e fornisce l'accesso ai dati.

COS'E' MVC?

E' un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica di business.

Il framework Laravel in PHP è basato sul pattern MVC e ci spinge a sviluppare la nostra applicazione con questo pattern.

COSE E' COMPOSTO IL MVC?

Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:
1) Model: fornisce i metodi per accedere ai dati utili all'app
2) View: visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti
3) Controller: riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti

COME FUNZIONA MVC IN LARAVEL?

1) Routing - app/Http/routes.php

Dall'URI a cui rispondere al Controller da chiamare

2) Controllers - app/Http/Controllers/

Ogni controller gestire il suo dominio applicativo, interroga il o i Model e ritorna una view passando dei dati

3) Model - app/

Ogni model si occupa di gestire le interazioni con i dati (DB)

4) View - resources/views/

La view è l'equivalente di una pagina HTML ma resa dinamica

ESEMPI DI ROUTING IN LARAVEL

1) Routing - app/Http/routes.php

```
Route::get('/', function () {
  return view('welcome');
});
Route::get('/', 'WelcomeController@index');
Route::get('home', 'HomeController@index');
Route::resource('articles', 'ArticlesController');
```

ESEMPI DI CONTROLLER IN LARAVEL

2) Controllers - app/Http/Controllers/
Dalla shell, eseguire il comando:
php artisan make:controller WelcomeController --plain
Apriamo il file app/Http/Controllers/WelcomeController.php
ed aggiungiamo il seguente metodo:

```
public function index() {
  return 'Hello World!';
}
```

ESEMPI DI CONTROLLER IN LARAVEL (PARTE 2)

Proviamo ora a sostituire il metodo appena creato con:

```
public function index() {
  $name = 'Leonardo Dal Zovo';
  return view('welcome.hello')- >with('name', $name);
}
```

oppure con più parametri:

```
return view('welcome.hello')->with(['name' => 'Leonardo', 'lastname' => 'Dal Zovo']);
```

altrimenti

```
return view('welcome.hello', $phpArray);
```

ESEMPI DI VIEW IN LARAVEL

4) View - resources/views/

Creiamo ora il file resources/views/welcome/hello.blade.php
Inseriamo il nostro codice HTML come abbiamo imparato ma ora aggiungiamo anche una delle seguenti righe:

```
<p>Hello <?= $name; ?> </p>
<p>Hello <? echo $name; ?> </p>
<p>Hello  {{ $name; }} </p>
```

GRAZIE

per l'attenzione!