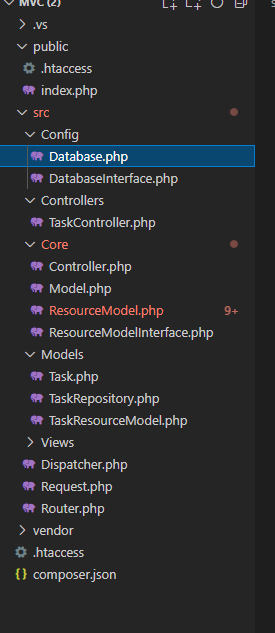
**Xây dựng PHP framework cơ bản theo mô hình MVC**

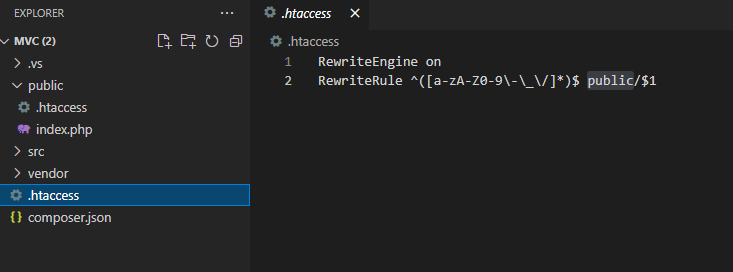
1. Luồng request.

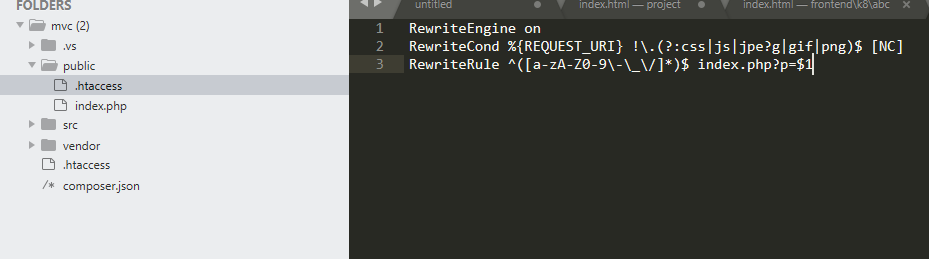


* Khi nhận request từ khách hàng, thì mặc vào file index.html hoặc index.php đầu tiên.
* File .htaccess

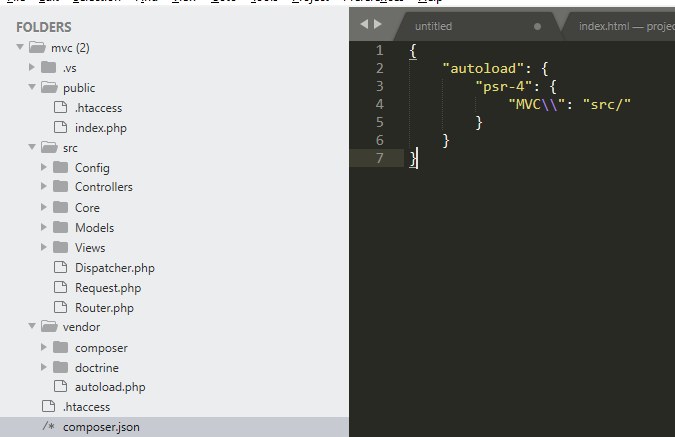
+ File .htaccess (hypertext access) là một file có ở thư mục gốc của các hostting và do apache quản lý, cấp quyền. File .htaccess có thể điều khiển, cấu hình được nhiều thứ với đa dạng các thông số, nó có thể thay đổi được các giá trị được set mặc định của apache.

Nếu được khai thác tốt thì .htaccess sẽ giúp được bạn rất nhiều việc mà tốn rất ít công sức đơn giản chi vài dòng lệnh là được. Cách thức mà bạn sử file .htaccess cũng thật đơn giản cứ mở bằng một trình soạn thảo nào đó chỉnh sửa xong save as lại với thành file .htaccess là xong.

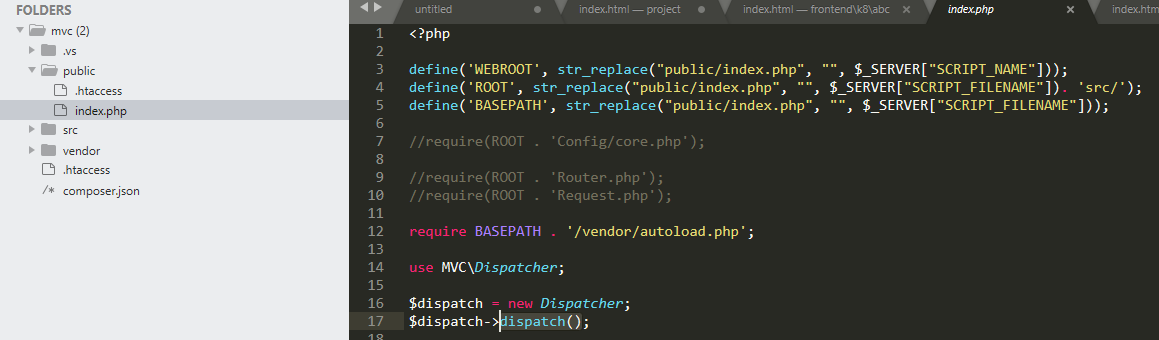
* Chuyển domain vào cấp thư mục public bằng RewriteRule
* Bỏ đuôi mở rộng của file web “.php”:

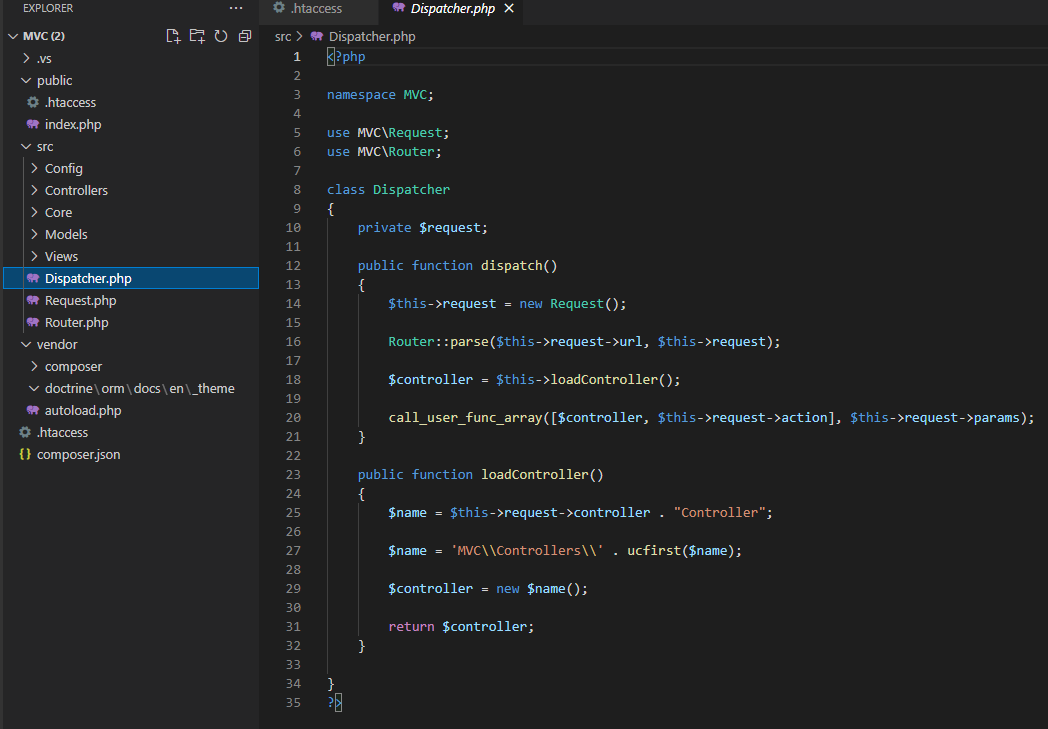


* Định nghĩa hằng số toàn cục với define
* require BASEPATH . '/vendor/autoload.php'; của psr-4.
* Tiếp theo, chúng ta có file composer.json: với Prefix MVC và thư mục root src/

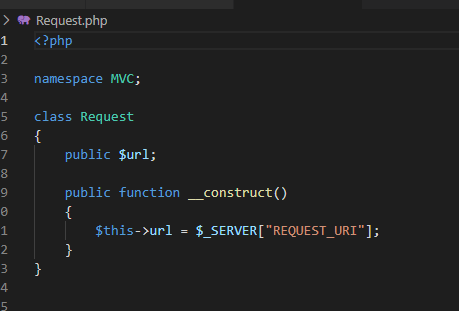


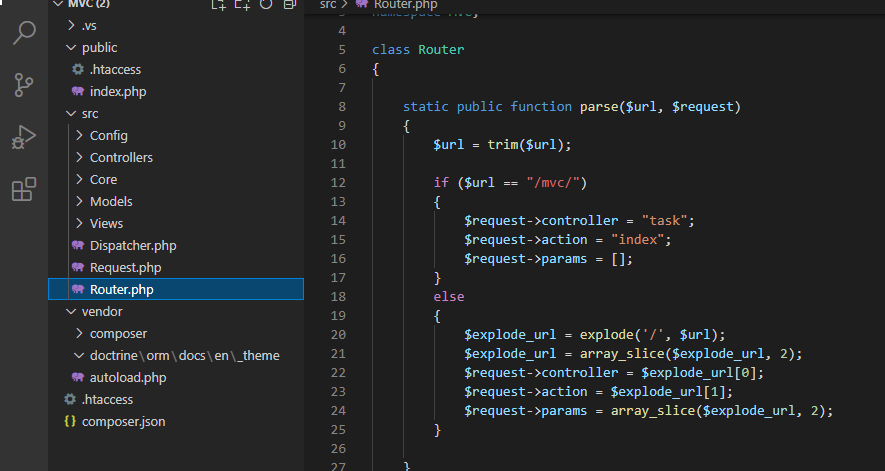
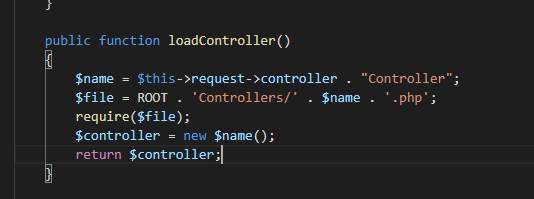
* Use và khởi tạo object Dispatcher , call method dispatch();





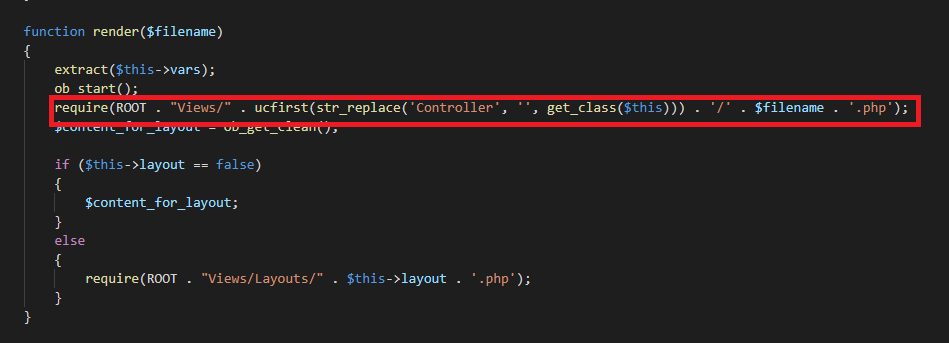
* Khởi tạo object Request



* Trong Request dùng \_\_construct() để gán $\_SERVER[‘REQEST\_URI’] cho thuộc tính $url
* Router::parse($this->request->url, $this->request);
* Trong Dispatcher Truyền url vừa nhận được và object request vào method static parse của Router
* Ở đây sẽ băm $url /root/controller/acction/params thành controller, acction và prams rồi gán thành các thuộc tính của object $request.
* Gọi function loadController
* Fuction sẽ return về object controler theo controler vừa lấy được trên url và gán vào request->controler.
* Tuy nhiên ở đây không sử dụng prs-4, các bạn phải chuyển lại thành psr-4 theo cấu trúc ví dụ như.
* $controller = “MVC\\ \Controllers\\TaskController”;
* return new $controller;
* Điều chỉnh Task là động theo url nhập vào.
* 
* Gọi function của controller theo đối số vừa lấy được
* call\_user\_func\_array([$controller, $this->request->action], $this->request->params);

Lưu ý cần đổi lại đường dẫn cho đúng vào view cần thiết

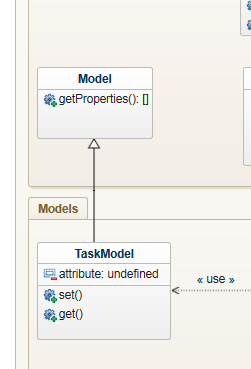
Trong file core/Controller



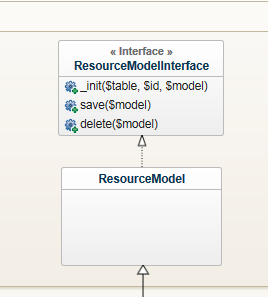
1. Tạo CRUD ( Các bạn triển khai thành bài hoàn chỉnh với gợi ý sau

và thêm code, try catch…. nếu cần thiết )

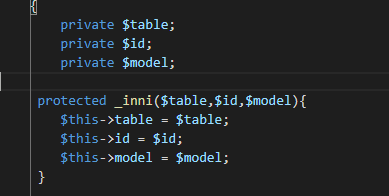
* Tạo class Model trong thư mục core với hàm getProperties($object) để chuyển đổi object sang mảng key-value với hàm get\_object\_vars($object).
* Tạo class TaskModel Trong thư mục Models extends từ class Model
* Dùng design pattern ORM để tạo model TaskModel ánh xạ với bảng task trong database và hàm get set.



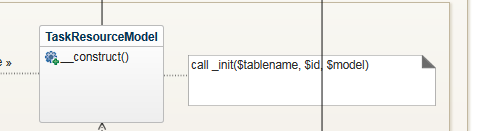
* Tạo inteface ResourceModelInterface trong thư mục core với method \_init, save, delete
* Tạo class ResourceModel trong thư mục model triển khai từ ResourceModelInterface. Resource model: Làm việc trực tiếp với database (CRUD).



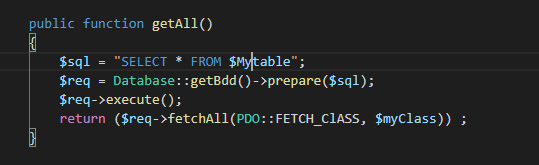
* Vì class ResourceModel trong core dùng chung cho các ResourceModel con nên ta chưa biết tên bảng, tên khóa chính và Model được dùng. Vì vật ta dùng \_init($table, $id, $model) để lấy tên bảng và các dữ liệu khác.



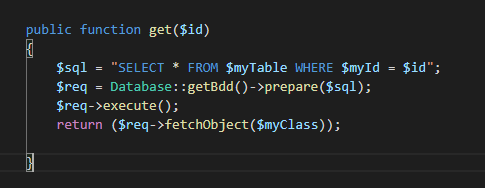
* Ở các ResourceModel con ví dụ TaskResourceModel ta chỉ cần gọi hàm \_innit và truyền đối số vào.
* class TaskResourceModel extends ResourceModel
* {
* public \_\_construct(){
* \_inni("tasks","taskId",new TaskModel);
* }
* }



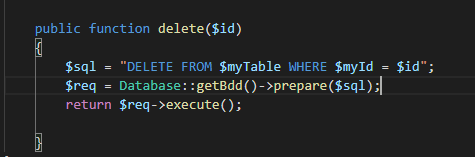
* Ở ResourceModel trong core thêm 4 hàm chính get($id), getAll(), save(), delete()
* Hàm getAll, các bạn chuyển myTable và myClass (gợi ý: có thể dùng hàm get\_class($this->model)) được nhận bằng hàm \_init



* Hàm get, các bạn chuyển myTable,myId, myClass được nhận bằng hàm \_init



* Hàm delete các bạn chuyển myTable,myId được nhận động bằng hàm \_init



* Hàm save($model) myTable,myId, myClass được nhận bằng hàm \_init

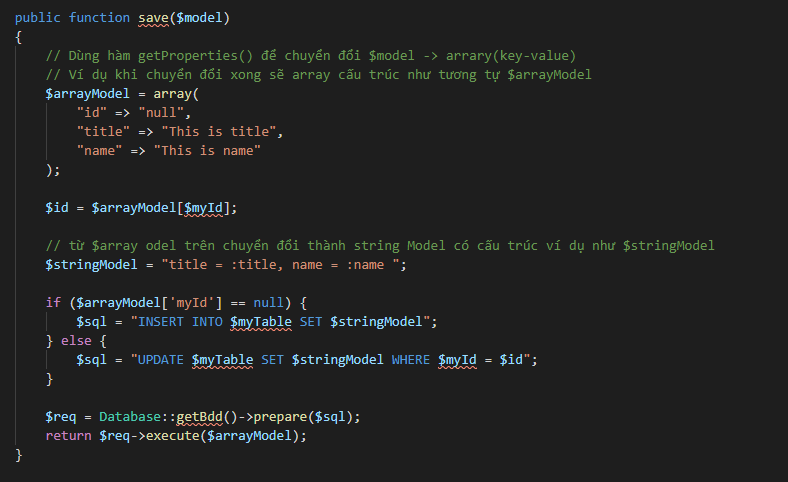
Chúng ta đổi object $model thành array (key-value) với hàm getProperties(). Ví dụ $arrayModel.

* Có 2 thuộc tính hợp sảy ra
* Khi $arrayModel[‘myId’] == null thì hàm là hàm insert với

$sql = INSERT INTO $myTable SET column1=: column1, column2=: column2,…;

* Khi $arrayModel[‘myId’] != null thì hàm là hàm insert với

$sql = UPDATE $myTable SET column1= :column1, column2= :column2,... WHERE $myId = $id



* Class TaskRepository sẽ khởi tạo TaskResourceModel theo design pattern Repository;
* Trong TaskController extends Controller sẽ khởi tạo và sử dụng các hàm trong TaskRepository
* TaskControler sẽ lấy các request cần thiết từ view ($\_POST, $\_GET….)
* Chuyển đổi sang $id Hoặc $TaskModel(bằng set()) rồi truyền đối số đó vào cái hàm của TaskRepository, để tạo acction với databae hoặc lấy dữ liêu từ model - resourceModel rồi trả về View

