Phân tích bài Time trong giải cookie hân hoan

Loại hình kiểm thử White box

Các file cho bao gồm file Router.php, timeController.php, timeModel.php, index.php

Ảnh chụp file index.php



Giải thích code

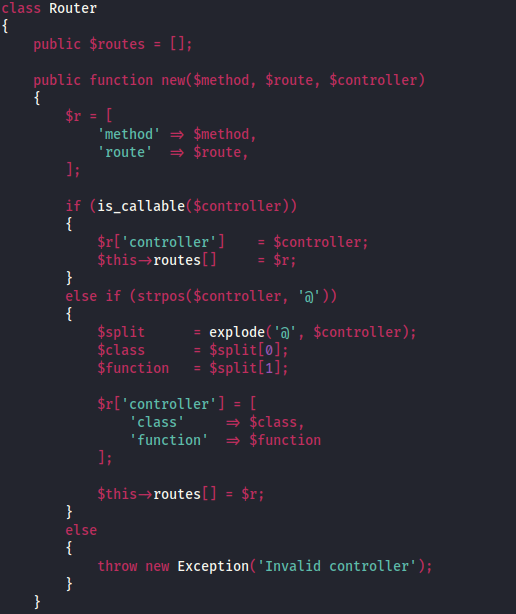
Sử dụng hàm spl\_autoload\_register để tải các lớp hoặc class khi cần thiết, kiểm tra với hàm preg\_match kiểm tra xem biến $name có kết thúc bằng Controllers không ví dụ như là nếu $name là "UserController", sau đoạn mã trên, giá trị của $name sẽ được thay đổi thành "controllers/UserController". Trường hợp khác là nếu như kết thúc bằng Model thì biến $name sẽ nhận giá trị models/+$name

Include\_once chèn file một lần gọi tới file là $name.php

Khai báo một đối tượng tên là router có lớp Router, đối tượng router nhận giá trị như sau thứ nhất là GET phương thức chuyển dữ liệu dưới dạng GET, ‘/’(giả thuyết thư mục root hoặc là đường dẫn thư mục), một string có tên là ‘TimeController@index’

Biến response lưu giá trị của đối tượng router gọi hàm match()

Chúng ta sang file khác có tên là Router.php do file này khá là dài nên sẽ chụp từng phần và giải thích từng phần



Thiết lập đối tượng Router, đầu tiên là biến $routers là một mảng rỗng, tạo một hàm có tên là new đây là phương thức khởi tạo một đối tượng chứa tham số mình nghĩ nó khá là giống với một constructor với giá trị bên trong

Ta có một biến mới có tên là $r hoạt động như một hashmap với key và value như sau thứ nhất key ‘method’ nhận giá trị từ biến $method được truyền vào từ việc gọi hàm tương tự với biến route như vậy với hàm index.php ta có thể hiểu như là hiện tại biến $r có giá trị

$r = [

‘method’ = GET,

‘route’ = /,

];

Phần if tiếp theo ta có

if (is\_callable($controller))

{

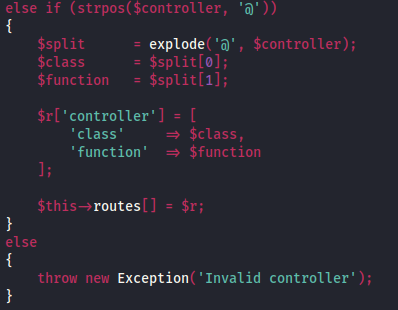
$r['controller'] = $controller;

$this->routes[] = $r;

}

Dùng hàm is\_callable với biến $controller từ giá trị gọi hàm kiểm tra xem liệu có thể gọi được hàm controller không thì ta có hashmap r với key là ‘controller’ nhận giá trị là biến controller đó và mảng routes sẽ được nhận hashmap $r

Nhưng nếu như không gọi được ta còn một trường hợp khác đó chính là



ở đây ta có nếu như controller mà chứa giá trị ‘@’ thì ta có thể hiểu như sau:

* $split sẽ hủy bỏ dấu @ đi với biến kiểm tra là $controller đang nhận giá trị là TimeController@index giờ thì biến split sẽ nhận hai giá trị là TimeController và index
* $class nhận giá trị là TimeController, $function nhận giá trị là index
* Hashmap $r với key là controller lại nhận thêm một hashmap nhỏ bên trong nữa giá trị hiện tại là

$r[‘controller’] = [

‘class’ = TimeController,

‘function’ = index

];

Tổng hợp lại ta được đến bây giờ hashmap là

$r = [

‘method’ = GET,

‘route’=’/’,

‘controller’=[

‘class’ = TimeController,

‘function’ = index

]

]

Trường hợp cuối nếu như không thỏa mãn cả hai trường hợp trên thì in ra một exception có tên là Invalid controller

Ta đến với hàm tiếp theo có tên là match()

A computer screen shot of a program code

Description automatically generated

Đây là một hàm không truyền vào tham số gì hết

Đầu tiên là một hàm lặp với mỗi giá trị có trong mảng routes được gọi là biến route

Nếu như hàm \_match\_route($route[‘route]) truyền vào với giá trị là value của mảng route với key là ‘route’ hiện tại giá trị này là ‘/’

Kiểm tra đợt 2 với value của key ‘method’ và request\_method của server khác nhau ta in ra thông báo 405

Biến $params nhận giá trị bằng việc gọi hàm getRouteParameters với tham số là value của hashmap $route với key là ‘route’ như vậy tham số truyền vào là ‘/’

Kiểm tra xem $route[‘controller’] có phải là một mảng hay không nếu có thì thực hiện việc lấy giá trị

* + $controller nhận giá trị từ $route[‘controller’]
  + $class nhận giá trị từ $controller[‘class’] giá trị hiện tại là TimeController
  + $function nhận giá trị từ $controller[‘function’] giá trị hiện tại là index

return (new $class)->$function($this, $params);

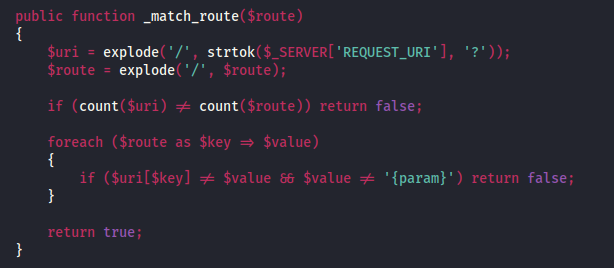
đoạn này có nghĩa là tạo một class mới tức là new TimeController gọi đến index($this, $params) với $this và $params là 2 tham số $this là routes

hoặc là hiểu như sau

return (new TimeController)->index(routes, $params)

còn đoạn ‘return $route['controller']($this,$params);’ có thể hiểu là với mỗi value trong key controller nhận giá trị từ $this và $params

Ta đến hàm tiếp theo



Hàm `\_match\_route` có nhiệm vụ kiểm tra xem đường dẫn của yêu cầu từ trình duyệt có khớp với mẫu đường dẫn của một tuyến đường (route) trong ứng dụng hay không.

Cụ thể, hàm thực hiện các bước sau:

1. Lấy đường dẫn yêu cầu từ biến môi trường `$\_SERVER['REQUEST\_URI']`.

2. Sử dụng hàm `strtok` để tách đường dẫn yêu cầu thành mảng các phần tử (thành phần giữa các dấu `/`).

3. Tách mẫu đường dẫn của tuyến đường (được truyền vào hàm) thành mảng các phần tử.

4. So sánh số lượng phần tử của đường dẫn yêu cầu và mẫu đường dẫn. Nếu khác nhau, có nghĩa là không khớp và hàm trả về `false`.

5. Lặp qua từng phần tử của mẫu đường dẫn và so sánh nó với phần tử tương ứng trong đường dẫn yêu cầu. Nếu chúng khác nhau và không có phần tử là `{param}` (được sử dụng để biểu thị tham số có thể thay đổi), thì hàm trả về `false`.

6. Nếu tất cả các so sánh trên đều khớp, tức là đường dẫn yêu cầu khớp với mẫu đường dẫn của tuyến đường, hàm trả về `true`.

Hàm `\_match\_route` giúp kiểm tra xem yêu cầu từ trình duyệt có khớp với các tuyến đường đã đăng ký trong ứng dụng hay không, dựa trên cấu trúc của đường dẫn và các mẫu đường dẫn đã được định nghĩa.

Hàm tiếp theo

A computer screen shot of a program code

Description automatically generated

Hàm `getRouteParameters` có nhiệm vụ trích xuất các tham số từ đường dẫn yêu cầu của trình duyệt, dựa trên một mẫu đường dẫn của tuyến đường đã đăng ký.

Cụ thể, hàm thực hiện các bước sau:

1. Khởi tạo một mảng rỗng `$params` để chứa các tham số trích xuất.

2. Lấy đường dẫn yêu cầu từ biến môi trường `$\_SERVER['REQUEST\_URI']` và tách nó thành mảng các phần tử (thành phần giữa các dấu `/`) bằng cách sử dụng hàm `explode`.

3. Tách mẫu đường dẫn của tuyến đường (được truyền vào hàm) thành mảng các phần tử.

4. Lặp qua từng phần tử của mẫu đường dẫn và so sánh nó với phần tử tương ứng trong đường dẫn yêu cầu:

- Nếu chúng khớp nhau, thì tiếp tục với phần tử tiếp theo (chấp nhận giống nhau, không cần trích xuất tham số).

- Nếu giá trị của phần tử trong mẫu đường dẫn là `{param}` và giá trị tương ứng trong đường dẫn yêu cầu không trống, thì thêm giá trị đó vào mảng `$params`.

- Nếu giá trị của phần tử trong mẫu đường dẫn là `{param}` và giá trị tương ứng trong đường dẫn yêu cầu trống, thì dừng vòng lặp và kết thúc với mã lỗi 404 (tham số bị thiếu).

5. Trả về mảng `$params` chứa các tham số đã trích xuất từ đường dẫn yêu cầu.

Hàm `getRouteParameters` được sử dụng để trích xuất các tham số từ đường dẫn yêu cầu, đồng thời kiểm tra và xử lý trường hợp khi tham số bị thiếu, giúp cung cấp dữ liệu cho việc xử lý tiếp theo của ứng dụng.

Hàm cuối cùng

A computer screen with text

Description automatically generated

Hàm `view` trong đoạn mã là một phần của hệ thống routing và controller của ứng dụng PHP. Nó có chức năng hiển thị các trang giao diện (views) cho người dùng, và đồng thời truyền dữ liệu từ controller đến view để hiển thị.

Dưới đây là giải thích từng phần của hàm `view`:

1. `view($view, $data = [])`: Đây là khai báo hàm, với hai tham số đầu vào. `$view` là tên của trang giao diện (view) mà bạn muốn hiển thị, còn `$data` là dữ liệu mà bạn muốn truyền vào view để hiển thị.

2. `extract($data)`: Hàm này thực hiện việc tách (extract) dữ liệu từ mảng `$data` và biến chúng thành các biến độc lập, dựa trên các khóa của mảng. Ví dụ: nếu bạn có `$data = ['name' => 'John', 'age' => 25]`, thì hàm `extract($data)` sẽ tạo hai biến `$name` và `$age` với các giá trị tương ứng.

3. `include \_\_DIR\_\_."/views/${view}.php";`: Đây là câu lệnh để chèn (include) một tệp giao diện (view) vào trang. Biến `$view` chứa tên của view cần hiển thị. Ví dụ, nếu `$view` là `'index'`, thì câu lệnh này sẽ chèn nội dung từ tệp `views/index.php`.

4. `exit;`: Câu lệnh này dừng việc thực thi chương trình ngay sau khi hiển thị view. Điều này đảm bảo rằng sau khi view được hiển thị, không có mã nguồn PHP nào khác được thực thi.

Tóm lại, hàm `view` có nhiệm vụ tạo một cách dễ dàng để hiển thị các trang giao diện trong ứng dụng PHP, đồng thời truyền dữ liệu từ controller tới view để hiển thị thông tin cho người dùng.

Giờ thì chúng ta sang hàm TimeController.php

A screen shot of a computer code

Description automatically generated

Hàm index với đối tượng router được truyền vào

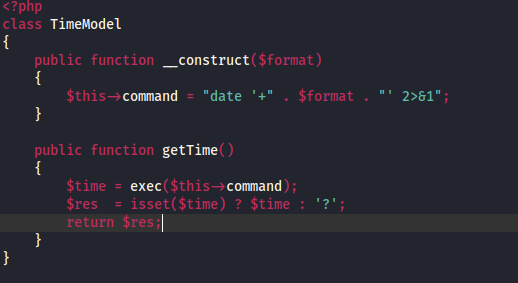
Biến $format lưu giá trị của biến format được truyền đi trong url dạng GET với format '%H:%M:%S'

$time là một đối tượng TimeModel có giá trị là $format

Trả về đối tượng router gọi hàm view chuyền vào giá trị ‘index’, hash map với time = $time gọi hàm getTime()

Giờ thì ta có thể hiểu rằng hàm view trong Router.php include \_DIR\_.”/view/index.php”

Ta đến file cuối cùng trong các file backend đó chính là file TimeModel.php



Đầu tiên khởi tạo một đối tượng có tên là TimeModel

Dựng một constructor với biến format được truyền vào

* Biến command lưu giá trị với kiểu date + format chẳng hạn như là date + ‘2023-08-04’

Hàm getTime

Biến $time thực hiện biến command đó