My Encounter, Analysis, and Solution to The TokenMismatchException Problem in Laravel 5.*

1. My Code

Here is the laravel login process code:

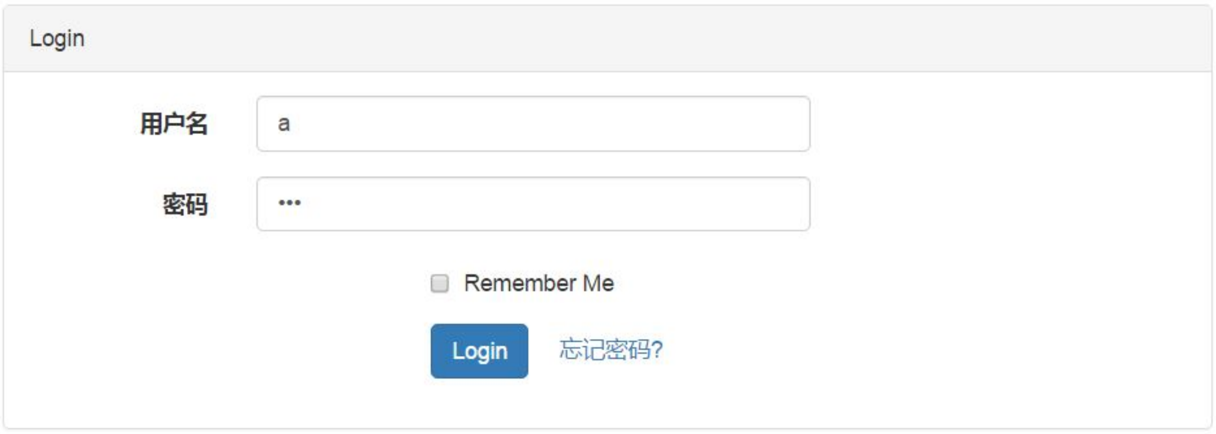
```
<form class="form-horizontal" role="form" method="POST" action="/auth/login">
                    <input type="hidden" name="_token" value="{{ csrf_token() }}">
```

And another ajax code of transfering data:

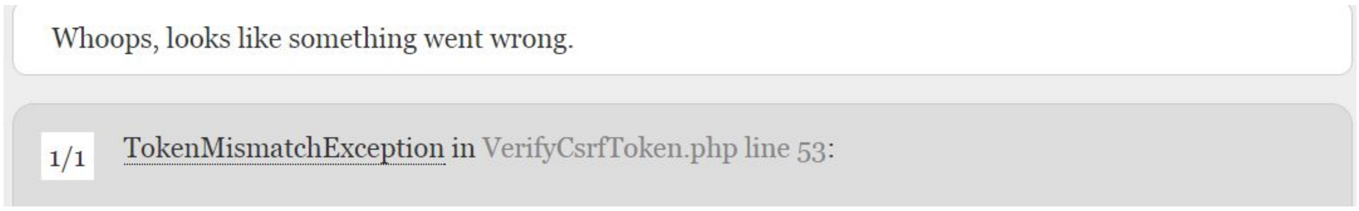
```
 $(document).ready(function(){
    $('.send-form-ajax').submit(function(event){
      $('.send-btn').fadeOut(300);
      event.preventDefault();
      $.ajax({
        url: 'home/editUser',
        type: "post",
        data: { 'name':$('input[name="name"]').val(),
                'company':$('input[name="company"]').val(),
                'department':$('input[name="department"]').val(),
                'isLocked':$('input[name="isLocked"]').is(':checked'),
                '_token':$('input[name="_token"]').val()
               },
        success: function(data){
            alert('ajax 请求成功');
        },
        error: function(e) {
            //called when there is an error
            //console.log(e.message);
            alert('ajax 请求失败');
        }
      });
    });
});
```

2. The Error

I use the Laravel 5.1 framework to build an application to manage my IP resources. While I get to authentication process, I encount the annoying TokenMismatch Exception. Below is the Exception:



While clicking the login button, the exception appears.

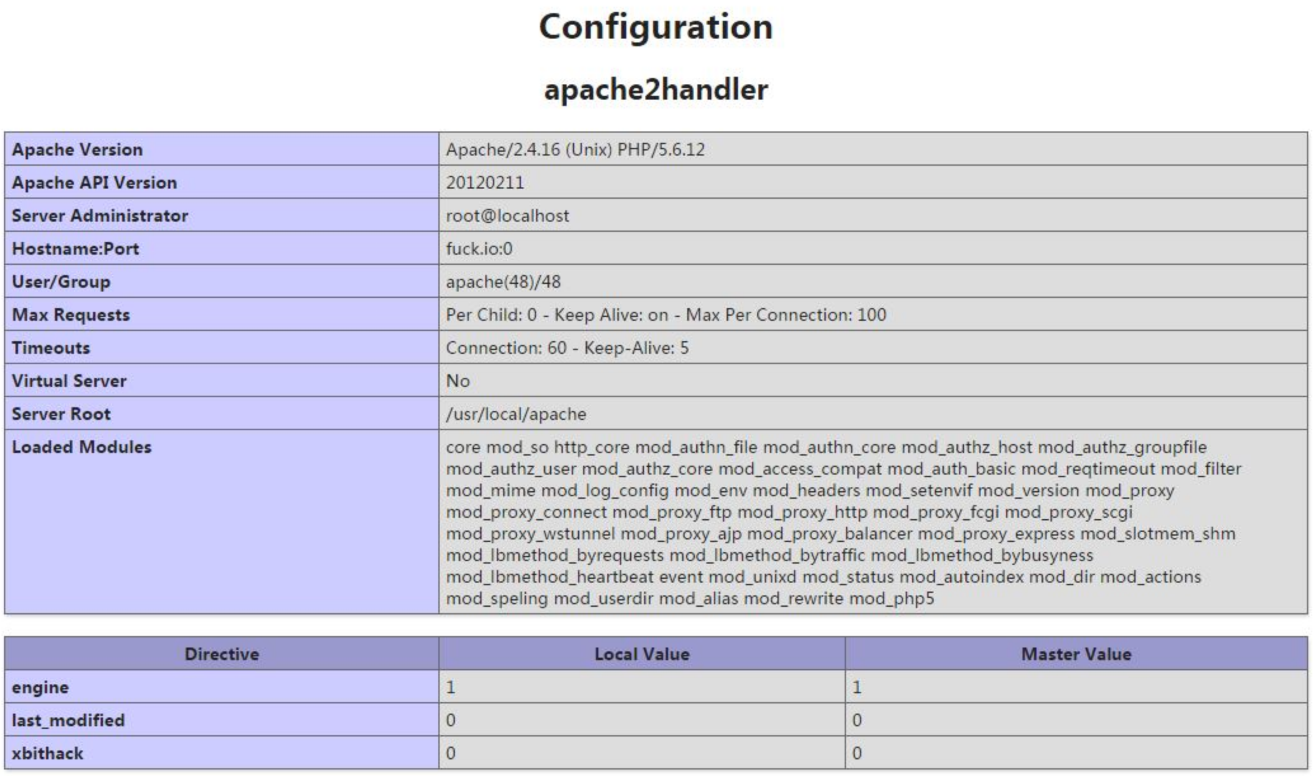


Whoops, looks like something went wrong.

1/1 TokenMismatchException in VerifyCsrfToken.php line 53:

3. The Environment

To help others locate their problem, following is my Server configuration.

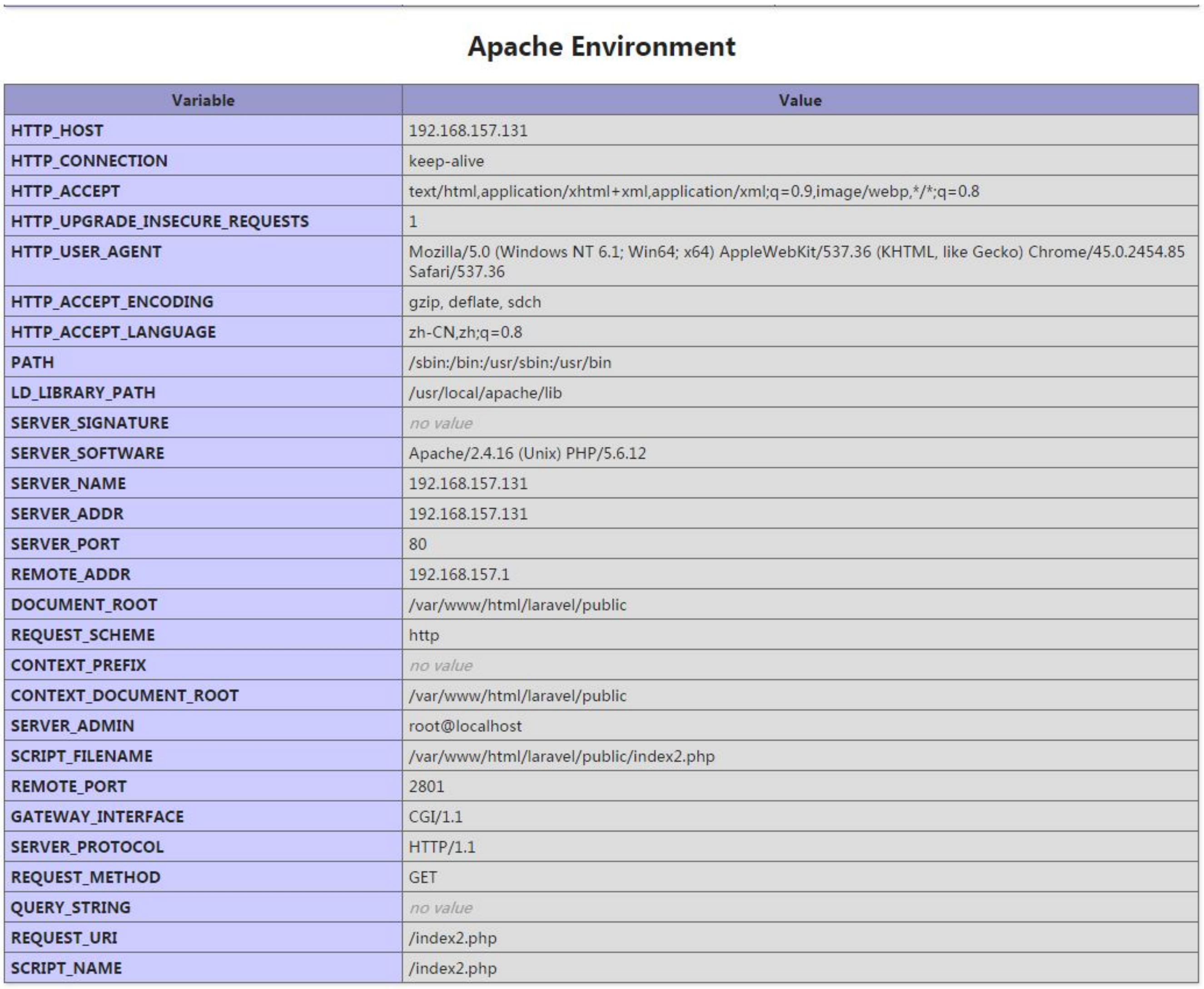


Configuration

apache2handler

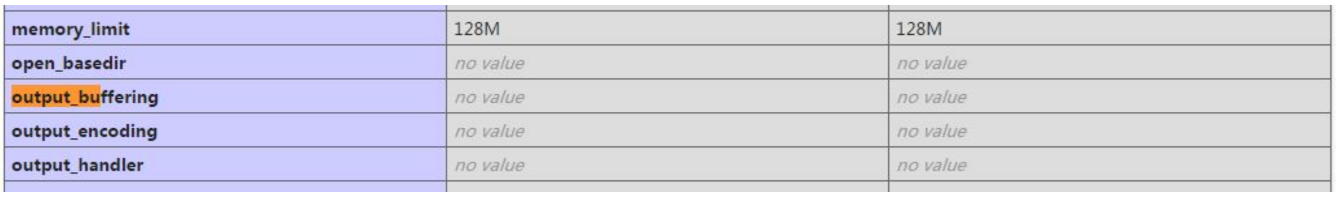
Apache Version	Apache/2.4.16 (Unix) PHP/5.6.12
Apache API Version	20120211
Server Administrator	root@localhost
Hostname:Port	fuck.io:0
User/Group	apache(48)/48
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 60 - Keep-Alive: 5
Virtual Server	No
Server Root	/usr/local/apache
Loaded Modules	core mod_so http_core mod_authn_file mod_authn_core mod_authz_host mod_authz_groupfile mod_authz_user mod_authz_core mod_access_compat mod_auth_basic mod_reqtimeout mod_filter mod_mime mod_log_config mod_env mod_headers mod_setenvif mod_version mod_proxy mod_proxy_connect mod_proxy_ftp mod_proxy_http mod_proxy_fcgi mod_proxy_scgi mod_proxy_wstunnel mod_proxy_ajp mod_proxy_balancer mod_proxy_express mod_slotmem_shm mod_lbmethod_byrequests mod_lbmethod_bytraffic mod_lbmethod_bybusyness mod_lbmethod_heartbeat event mod_unixd mod_status mod_autoindex mod_dir mod_actions mod_speling mod_userdir mod_alias mod_rewrite mod_php5

Directive	Local Value	Master Value
engine	1	1
last_modified	0	0
xbithack	0	0



Apache Environment

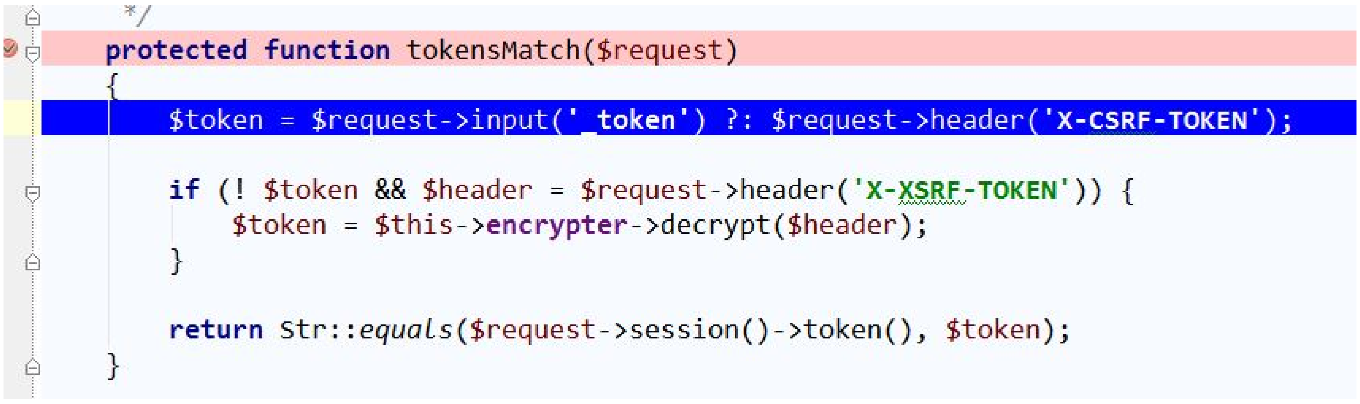
Variable	Value
HTTP_HOST	192.168.157.131
HTTP_CONNECTION	keep-alive
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
HTTP_UPGRADE_INSECURE_REQUESTS	1
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.85 Safari/537.36
HTTP_ACCEPT_ENCODING	gzip, deflate, sdch
HTTP_ACCEPT_LANGUAGE	zh-CN,zh;q=0.8
PATH	/sbin:/bin:/usr/sbin:/usr/bin
LD_LIBRARY_PATH	/usr/local/apache/lib
SERVER_SIGNATURE	*no value*
SERVER_SOFTWARE	Apache/2.4.16 (Unix) PHP/5.6.12
SERVER_NAME	192.168.157.131
SERVER_ADDR	192.168.157.131
SERVER_PORT	80
REMOTE_ADDR	192.168.157.1
DOCUMENT_ROOT	/var/www/html/laravel/public
REQUEST_SCHEME	http
CONTEXT_PREFIX	*no value*
CONTEXT_DOCUMENT_ROOT	/var/www/html/laravel/public
SERVER_ADMIN	root@localhost
SCRIPT_FILENAME	/var/www/html/laravel/public/index2.php
REMOTE_PORT	2801
GATEWAY_INTERFACE	CGI/1.1
SERVER_PROTOCOL	HTTP/1.1
REQUEST_METHOD	GET
QUERY_STRING	*no value*
REQUEST_URI	/index2.php
SCRIPT_NAME	/index2.php



memory_limit	128M	128M
open_basedir	*no value*	*no value*
output_buffering	*no value*	*no value*
output_encoding	*no value*	*no value*
output_handler	*no value*	*no value*

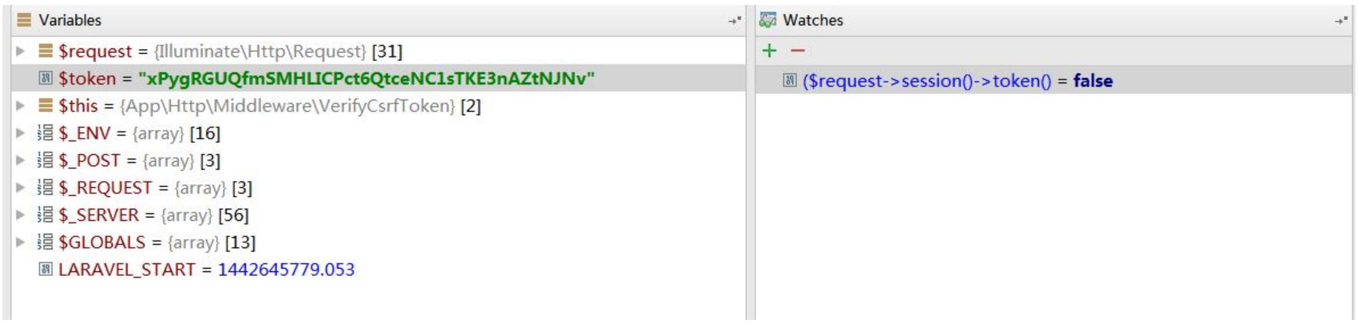
4. The Analysis

I directly go the VerifyCsrfToken.php to check what is happening there. I set a breakpoint and send a request from my browser.



```
    */
    protected function tokensMatch($request)
    {
        $token = $request->input('_token') ?: $request->header('X-CSRF-TOKEN');

        if (! $token && $header = $request->header('X-XSRF-TOKEN')) {
            $token = $this->encrypter->decrypt($header);
        }

        return Str::equals($request->session()->token(), $token);
    }
```

I watch the token transfered from request and session, and their are really not equated.



```
Variables
$request = {Illuminate\Http\Request} [31]
$token = "xPygRGUQfmSMHLICPct6QtceNC1sTKE3nAZtNJNv"
$this = {App\Http\Middleware\VerifyCsrfToken} [2]
$_ENV = {array} [16]
$_POST = {array} [3]
$_REQUEST = {array} [3]
$_SERVER = {array} [56]
$GLOBALS = {array} [13]
LARAVEL_START = 1442645779.053
```

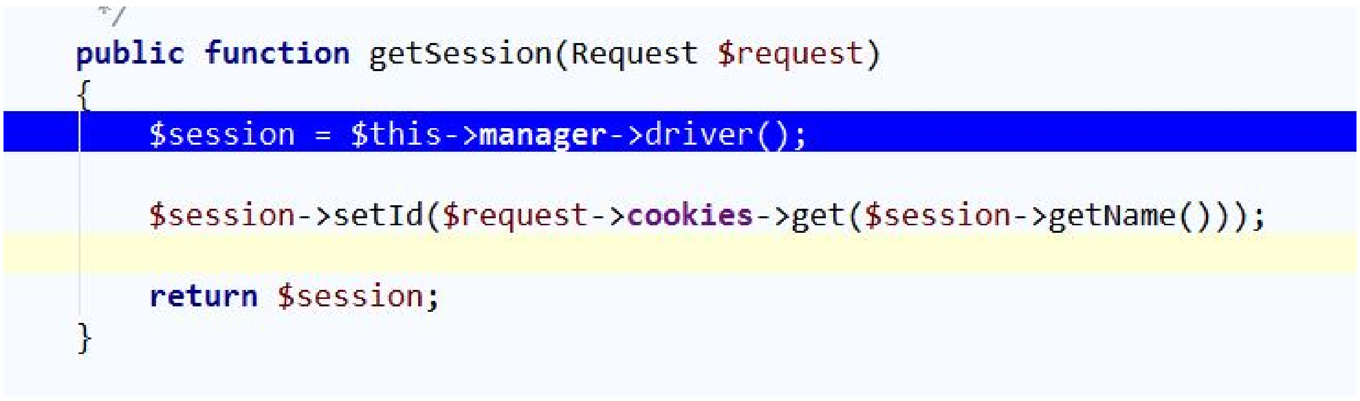
```
Watches
($request->session()->token() = false
```

The token of html is correctedly sent to the request.

```
<form class="form-horizontal" role="form" method="POST" action="/auth/login">
    <input type="hidden" name="_token" value="xPygRGUQfmSMHLICPct6QtceNC1sTKE3nAZtNJNv">
```

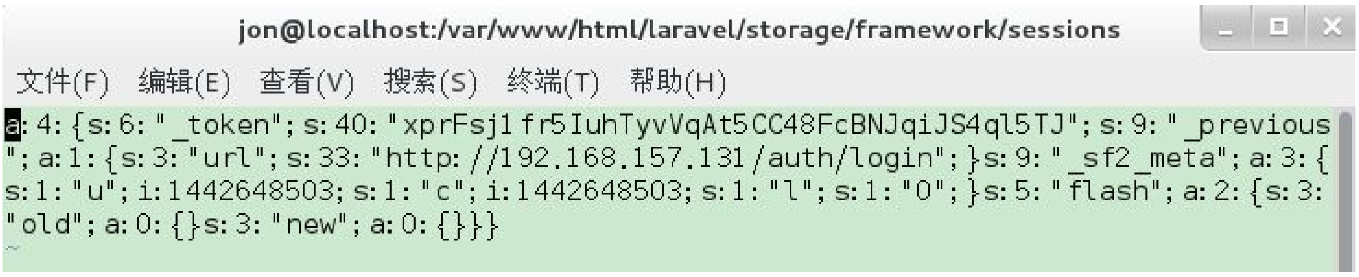
So the problem is actually caused by the session! We know that Laravel will start a new session if there is no cookie received. Here is how it's done.

```
public function handle($request, Closure $next)
{
    $this->sessionHandled = true;

    // If a session driver has been configured, we will need to start the session here
    // so that the data is ready for an application. Note that the Laravel sessions
    // do not make use of PHP "native" sessions in any way since they are crappy.
    if ($this->sessionConfigured()) {
        $session = $this->startSession($request);

        $request->setSession($session);
    }
```



```
 */
public function getSession(Request $request)
{
    $session = $this->manager->driver();

    $session->setId($request->cookies->get($session->getName()));

    return $session;
}
```

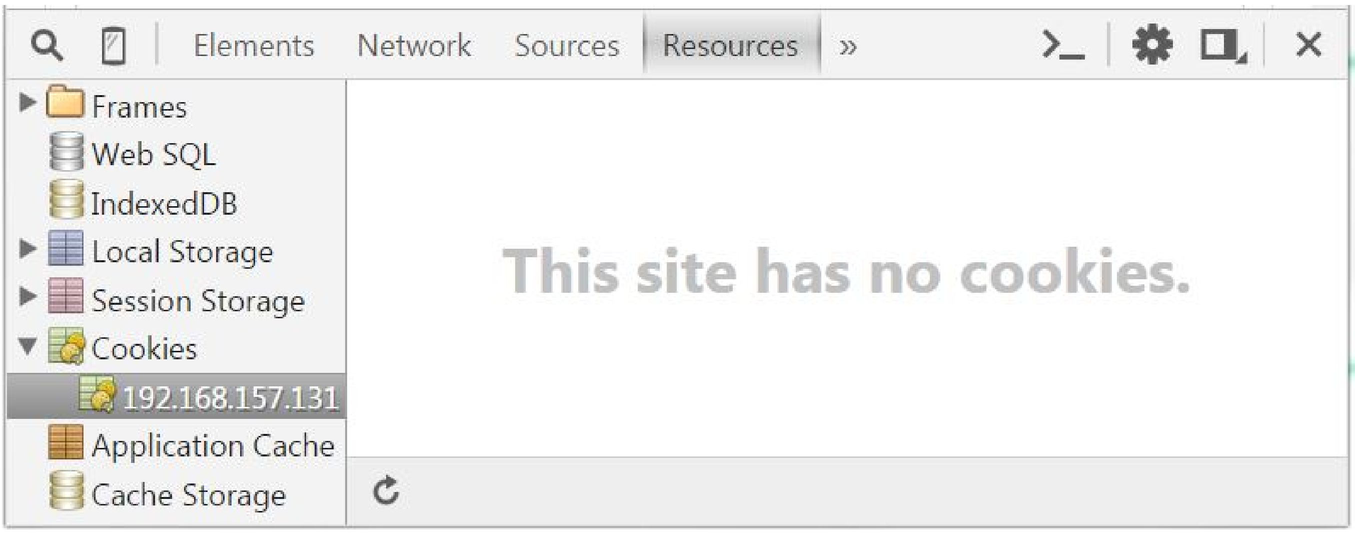
We can know from that the session tries to get cookies from request, while the cookie is sent from browser. The Server will jot down session information including CSRF token into files or database based on you drivers.



jon@localhost:/var/www/html/laravel/storage/framework/sessions

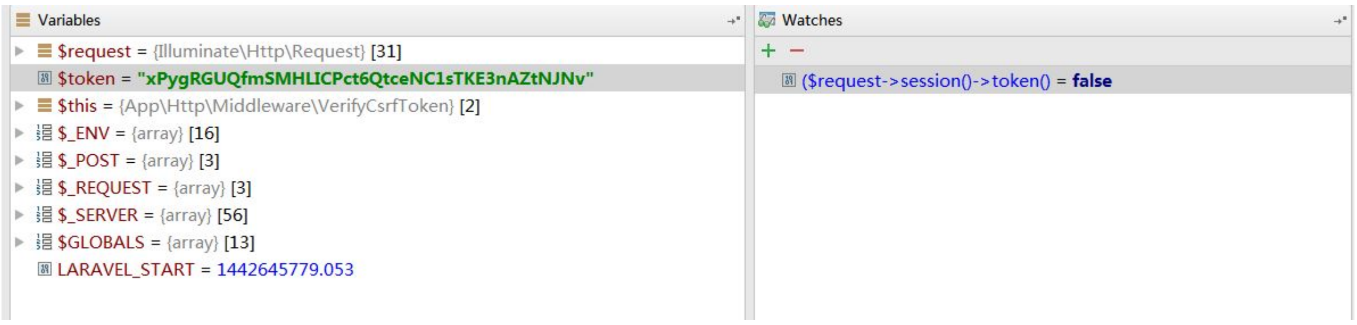
```
a:4:{s:6:"_token";s:40:"xprFsj1fr5IuhTyvVqAt5CC48FcBNJqiJS4ql5TJ";s:9:"_previous
";a:1:{s:3:"url";s:33:"http://192.168.157.131/auth/login";}s:9:"_sf2_meta";a:3:{
s:1:"u";i:1442648503;s:1:"c";i:1442648503;s:1:"l";s:1:"0";}s:5:"flash";a:2:{s:3:
"old";a:0:{}s:3:"new";a:0:{}}}
~
```

But my browser doesn't have any cookies to send:



This site has no cookies.

If Laravel doesn't get cookie from request, it will generate a new session ID and new CSRF token within next steps. Laravel addes these new cookies and headers to the response of the request, then send them to browser. But now the value of ($request->session()->token()) is false due to there is no value and new session has not been started yet.

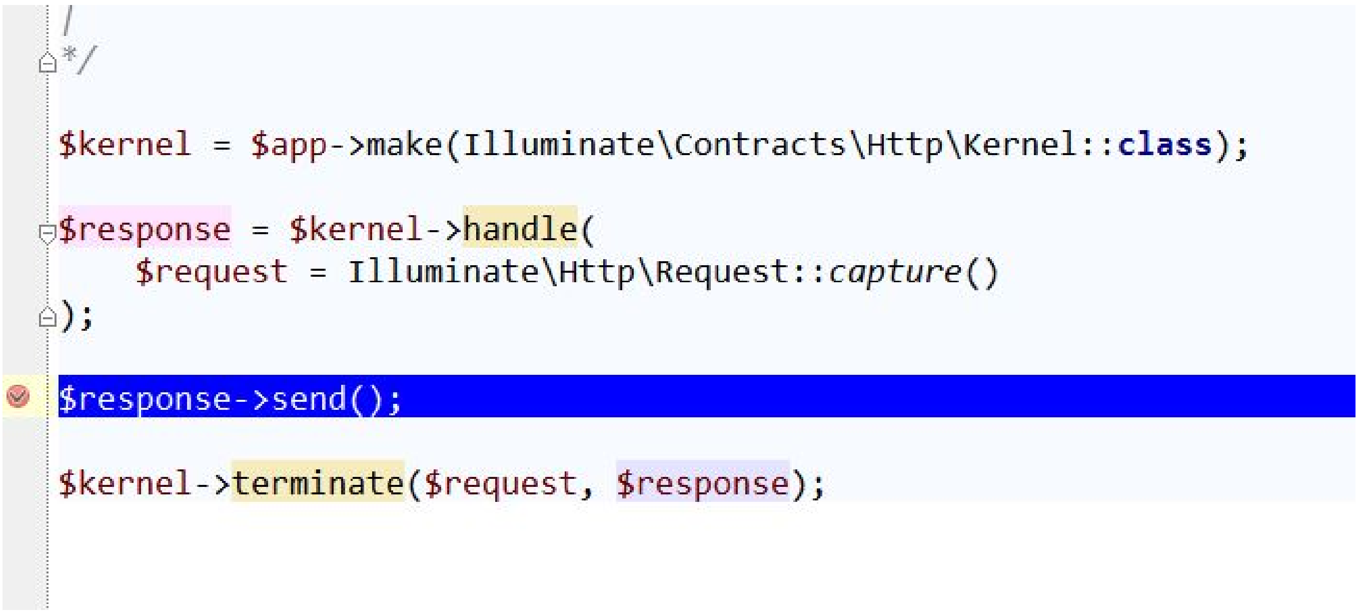


```
$request = {Illuminate\Http\Request} [31]
$token = "xPygRGUQfmSMHLICPct6QtceNC1sTKE3nAZtNJNv"
$this = {App\Http\Middleware\VerifyCsrfToken} [2]
$_ENV = {array} [16]
$_POST = {array} [3]
$_REQUEST = {array} [3]
$_SERVER = {array} [56]
$GLOBALS = {array} [13]
LARAVEL_START = 1442645779.053
```

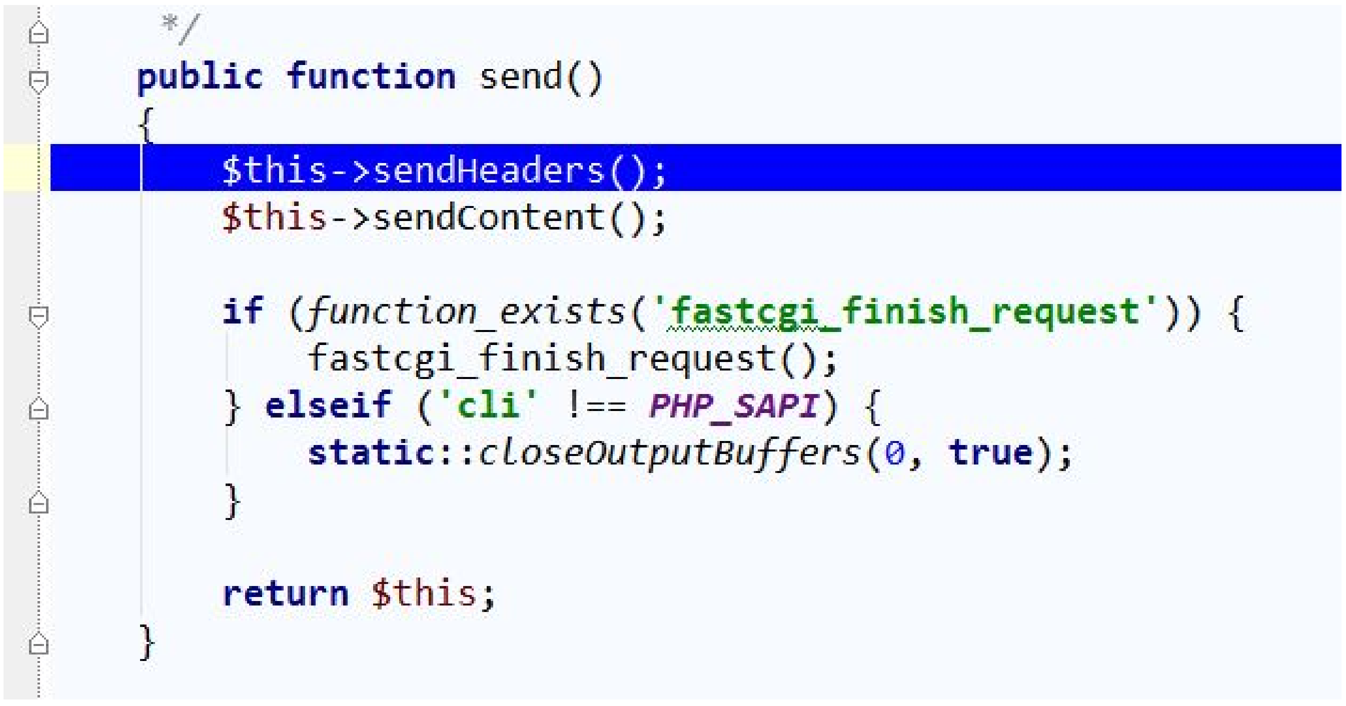
```
($request->session()->token() = false
```

Why doesn't the Server generate cookie and send to browser within precedent requests?

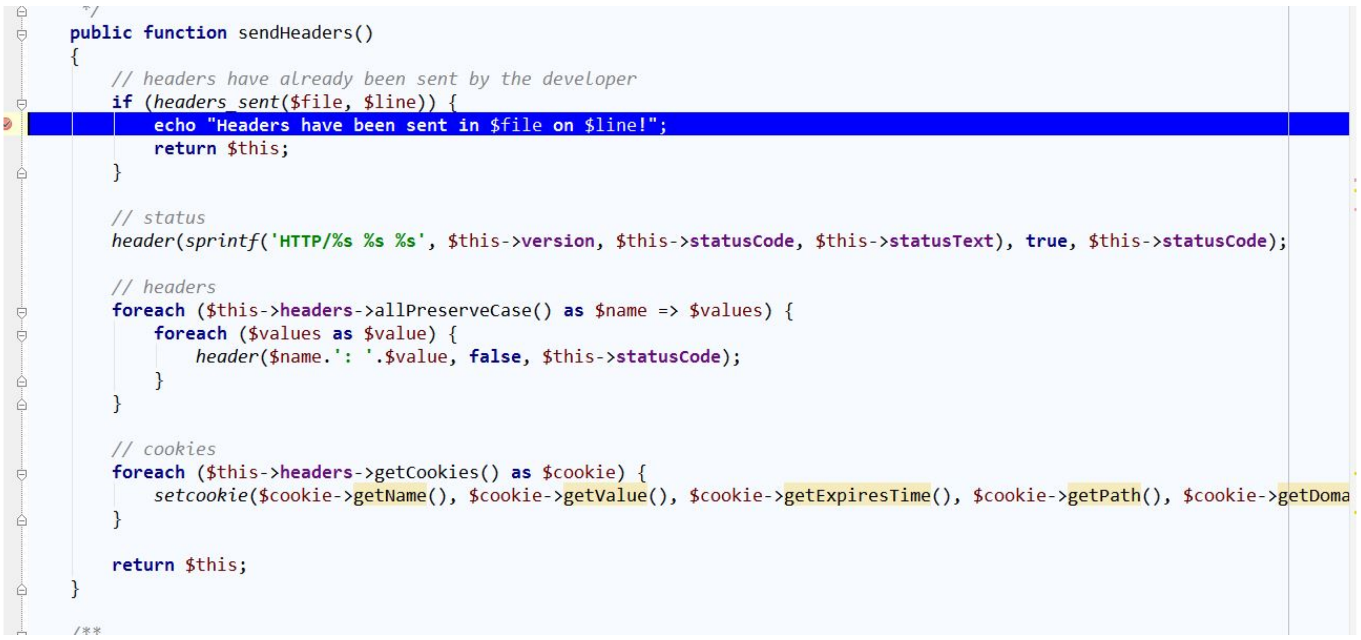
So we have to trace how the Laravel generates and sends cookie. The laravel generates response class within following processing request and then sends the response. We can know that from index.php.



```
|
*/

$kernel = $app->make(Illuminate\Contracts\Http\Kernel::class);

$response = $kernel->handle(
    $request = Illuminate\Http\Request::capture()
);

$response->send();

$kernel->terminate($request, $response);
```

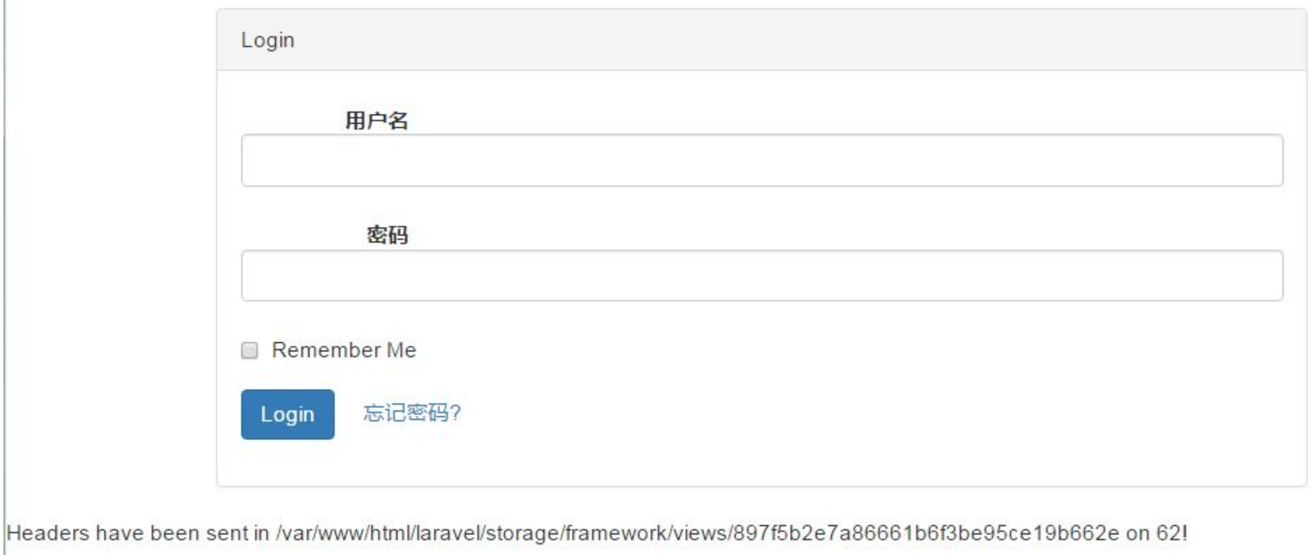


```
    */
    public function send()
    {
        $this->sendHeaders();
        $this->sendContent();

        if (function_exists('fastcgi_finish_request')) {
            fastcgi_finish_request();
        } elseif ('cli' !== PHP_SAPI) {
            static::closeOutputBuffers(0, true);
        }

        return $this;
    }
```



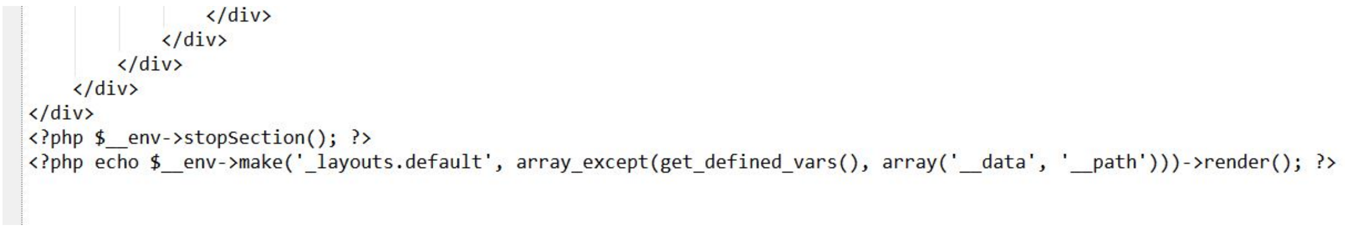
```
     */
    public function sendHeaders()
    {
        // headers have already been sent by the developer
        if (headers_sent($file, $line)) {
            echo "Headers have been sent in $file on $line!";
            return $this;
        }

        // status
        header(sprintf('HTTP/%s %s %s', $this->version, $this->statusCode, $this->statusText), true, $this->statusCode);

        // headers
        foreach ($this->headers->allPreserveCase() as $name => $values) {
            foreach ($values as $value) {
                header($name.': '.$value, false, $this->statusCode);
            }
        }

        // cookies
        foreach ($this->headers->getCookies() as $cookie) {
            setcookie($cookie->getName(), $cookie->getValue(), $cookie->getExpiresTime(), $cookie->getPath(), $cookie->getDoma
        }

        return $this;
    }

    /**
```

The result is that the Headers have already been sent, but I don't send any headers manually.



Headers have been sent in /var/www/html/laravel/storage/framework/views/897f5b2e7a86661b6f3be95ce19b662e on 62!

Since the information shows the file /var/www/html/laravel/storage/framework/views/897f5b2e7a86661b6f3be95ce19b662e has already done those creepy things, I checked the file. It is a compiled view file of outputing my login HTML code. It donesn't do anything like those.

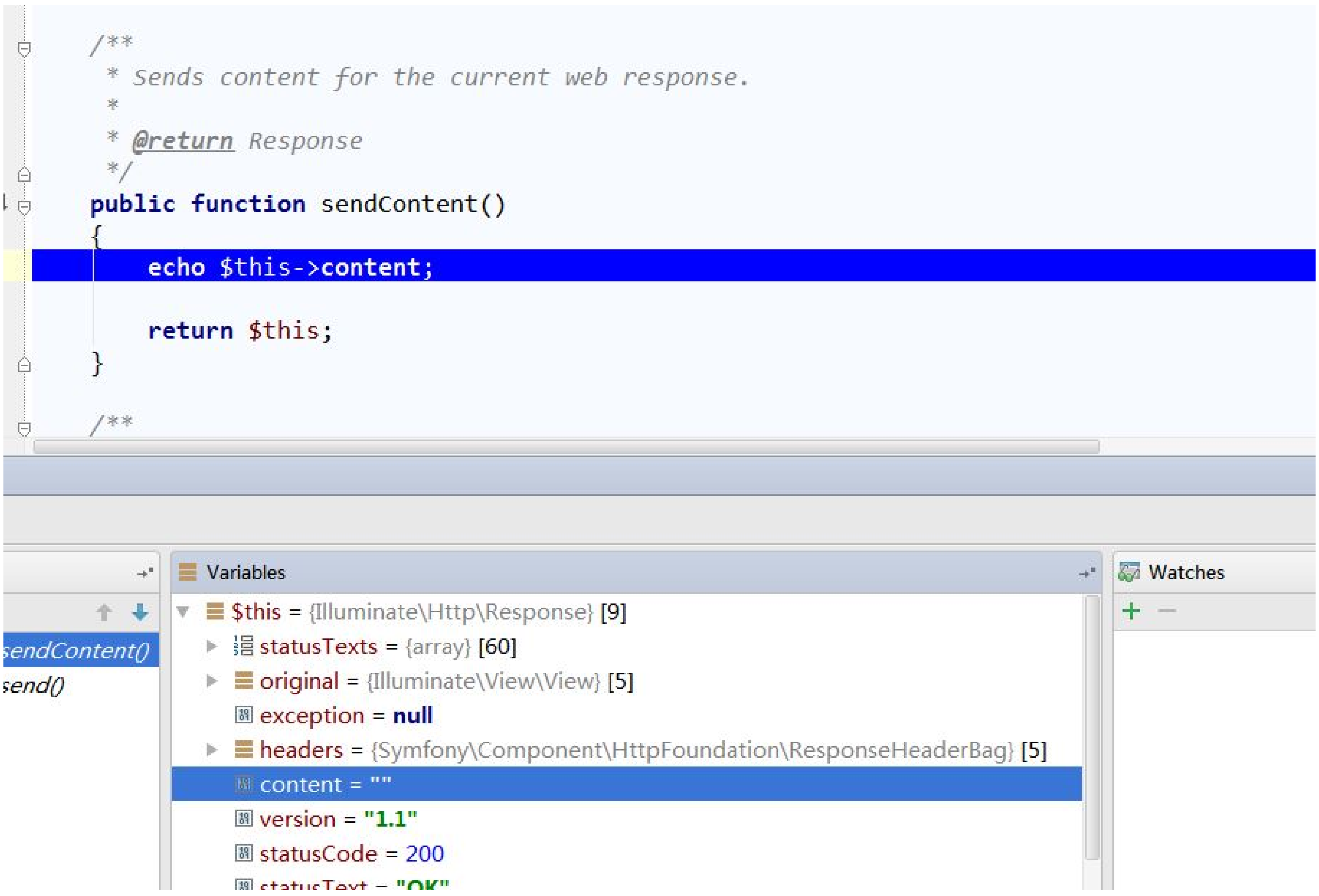


```
                </div>
            </div>
        </div>
    </div>
</div>
<?php $__env->stopSection(); ?>
<?php echo $__env->make('_layouts.default', array_except(get_defined_vars(), array('__data', '__path')))->render(); ?>
```

I really spend much times digging why the view file will send headers without any code like sending headers? I google the reason online and the two sites enlights me up. https://docs.joomla.org/Cannot*modify*header*information-headers*already_sent and http://digitalpbk.com/php/warning-cannot-modify-header-information-headers-already-sent

In the HTTP protocol a server response consists of a group of headers followed by a body, separated by a single blank line (i.e. a line containing only a carriage-return). This warning message is produced by PHP if a program attempts to send an additional HTTP header after the separator (and all the headers) has already been sent.

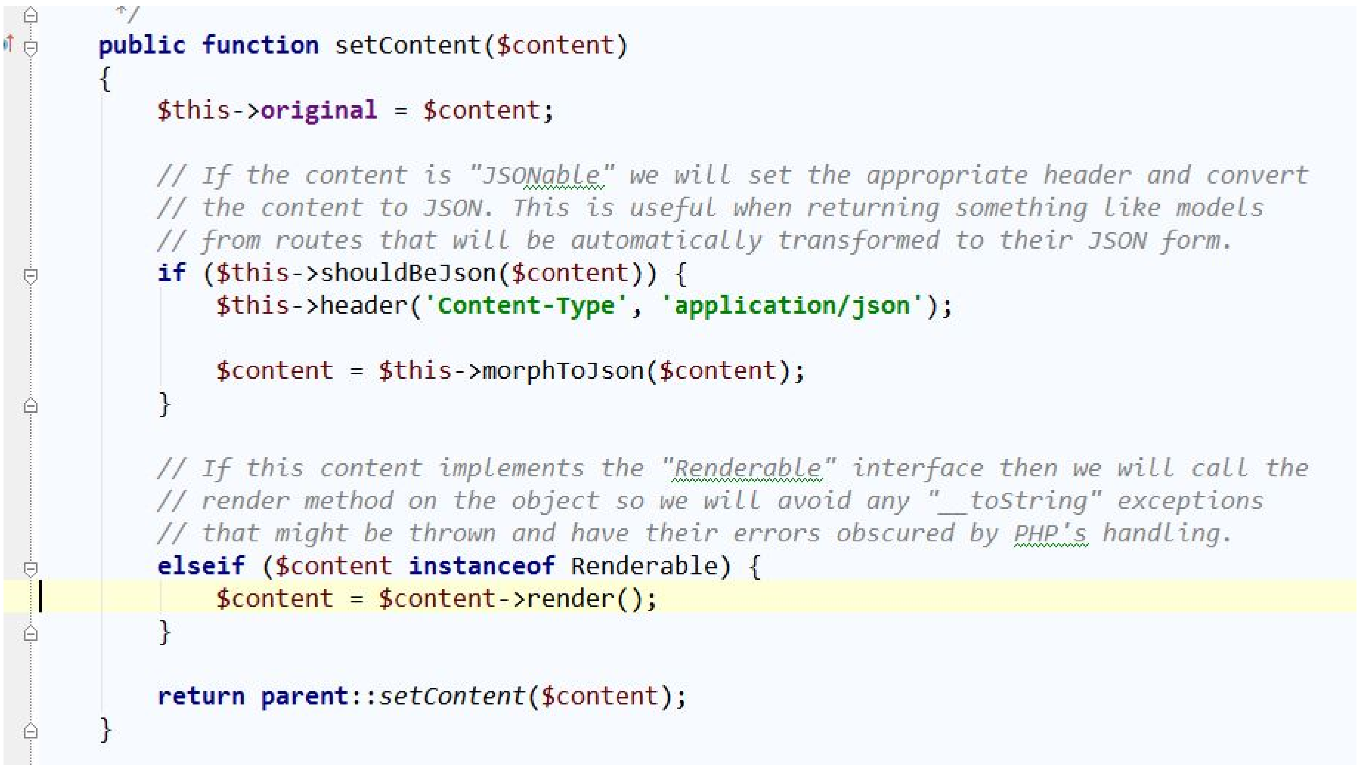
So I have to dig How the view was processed? Another interesting thing is that the content of the response is empty. If normal, the content will be HTML code string of view. If it is empty, why the site showes normally?



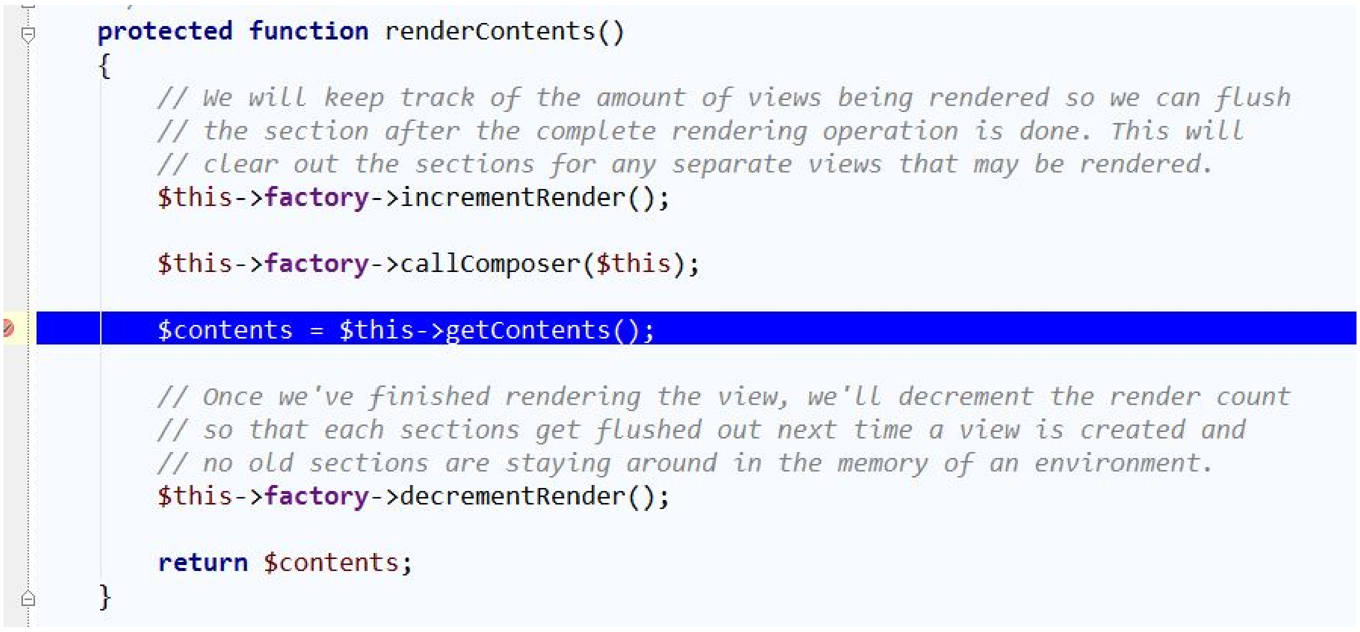
```
    /**
     * Sends content for the current web response.
     *
     * @return Response
     */
    public function sendContent()
    {
        echo $this->content;

        return $this;
    }

    /**
```

```
sendContent()
send()

Variables
$this = {Illuminate\Http\Response} [9]
    statusTexts = {array} [60]
    original = {Illuminate\View\View} [5]
    exception = null
    headers = {Symfony\Component\HttpFoundation\ResponseHeaderBag} [5]
    content = ""
    version = "1.1"
    statusCode = 200
```

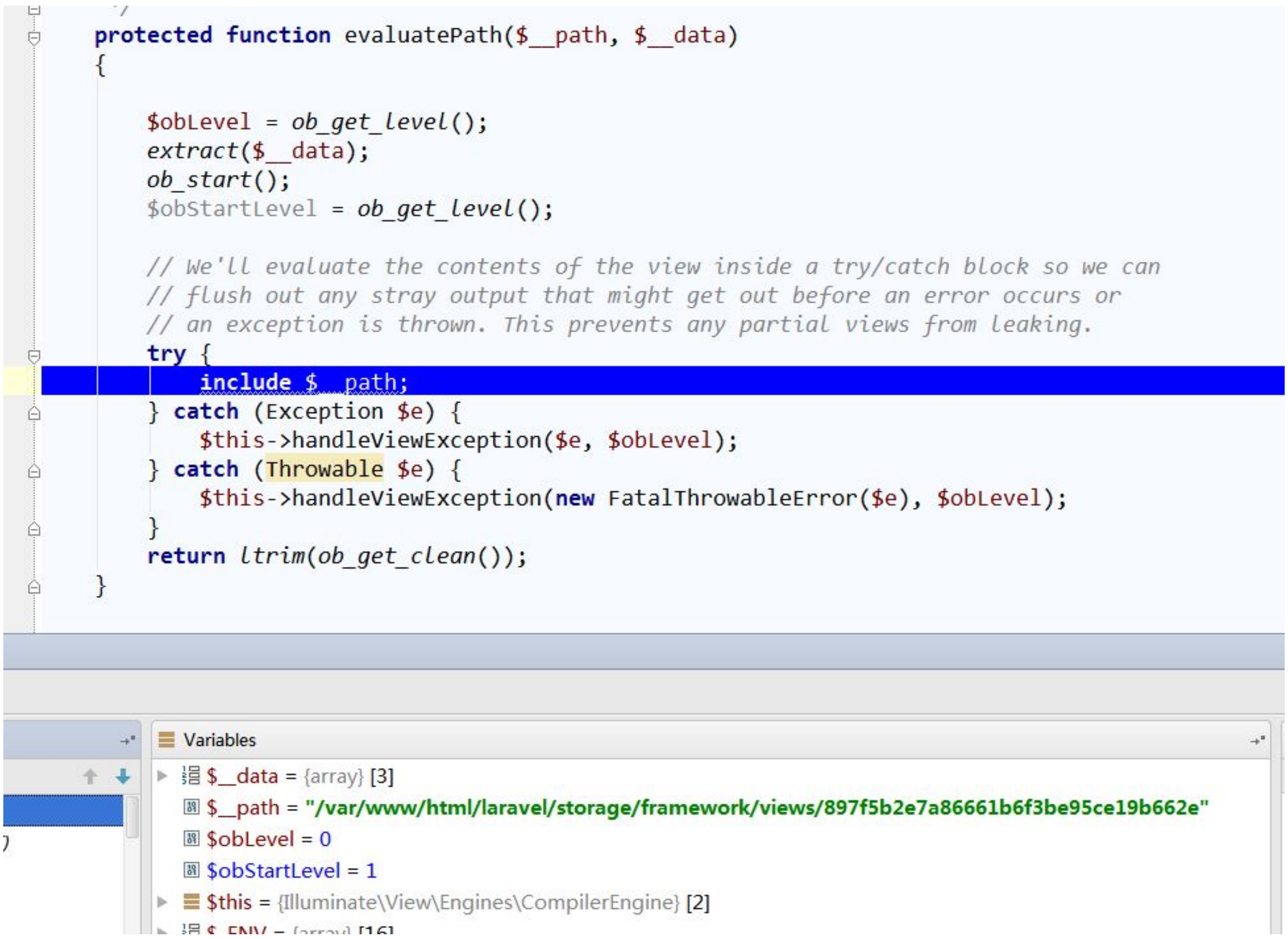
After a long track of the process of response and view, we finally get to the core code of dealing view.



```
     */
    public function setContent($content)
    {
        $this->original = $content;

        // If the content is "JSONable" we will set the appropriate header and convert
        // the content to JSON. This is useful when returning something like models
        // from routes that will be automatically transformed to their JSON form.
        if ($this->shouldBeJson($content)) {
            $this->header('Content-Type', 'application/json');

            $content = $this->morphToJson($content);
        }

        // If this content implements the "Renderable" interface then we will call the
        // render method on the object so we will avoid any "__toString" exceptions
        // that might be thrown and have their errors obscured by PHP's handling.
        elseif ($content instanceof Renderable) {
            $content = $content->render();
        }

        return parent::setContent($content);
    }
```



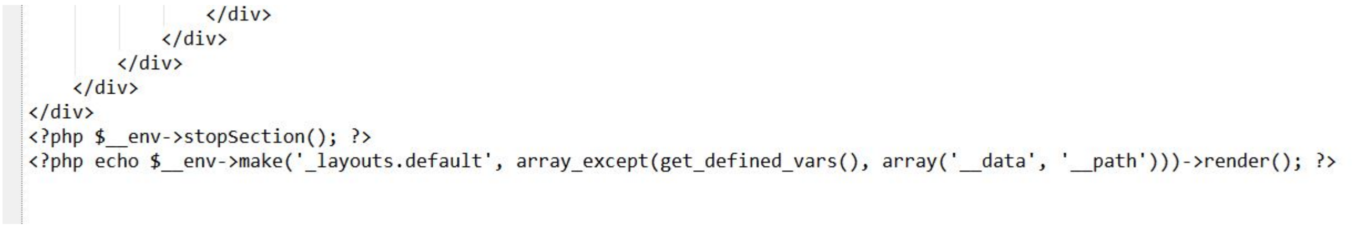
```
protected function renderContents()
{
    // We will keep track of the amount of views being rendered so we can flush
    // the section after the complete rendering operation is done. This will
    // clear out the sections for any separate views that may be rendered.
    $this->factory->incrementRender();

    $this->factory->callComposer($this);

    $contents = $this->getContents();

    // Once we've finished rendering the view, we'll decrement the render count
    // so that each sections get flushed out next time a view is created and
    // no old sections are staying around in the memory of an environment.
    $this->factory->decrementRender();

    return $contents;
}
```



```
protected function evaluatePath($__path, $__data)
{

    $obLevel = ob_get_level();
    extract($__data);
    ob_start();
    $obStartLevel = ob_get_level();

    // We'll evaluate the contents of the view inside a try/catch block so we can
    // flush out any stray output that might get out before an error occurs or
    // an exception is thrown. This prevents any partial views from leaking.
    try {
        include $__path;
    } catch (Exception $e) {
        $this->handleViewException($e, $obLevel);
    } catch (Throwable $e) {
        $this->handleViewException(new FatalThrowableError($e), $obLevel);
    }
    return ltrim(ob_get_clean());
}
```

```
Variables
$__data = {array} [3]
$__path = "/var/www/html/laravel/storage/framework/views/897f5b2e7a86661b6f3be95ce19b662e"
$obLevel = 0
$obStartLevel = 1
$this = {Illuminate\View\Engines\CompilerEngine} [2]
$_ENV = {array} [16]
```

We can find "include $**path" in the view rendering. while the variable $**path is compiled view of current site.

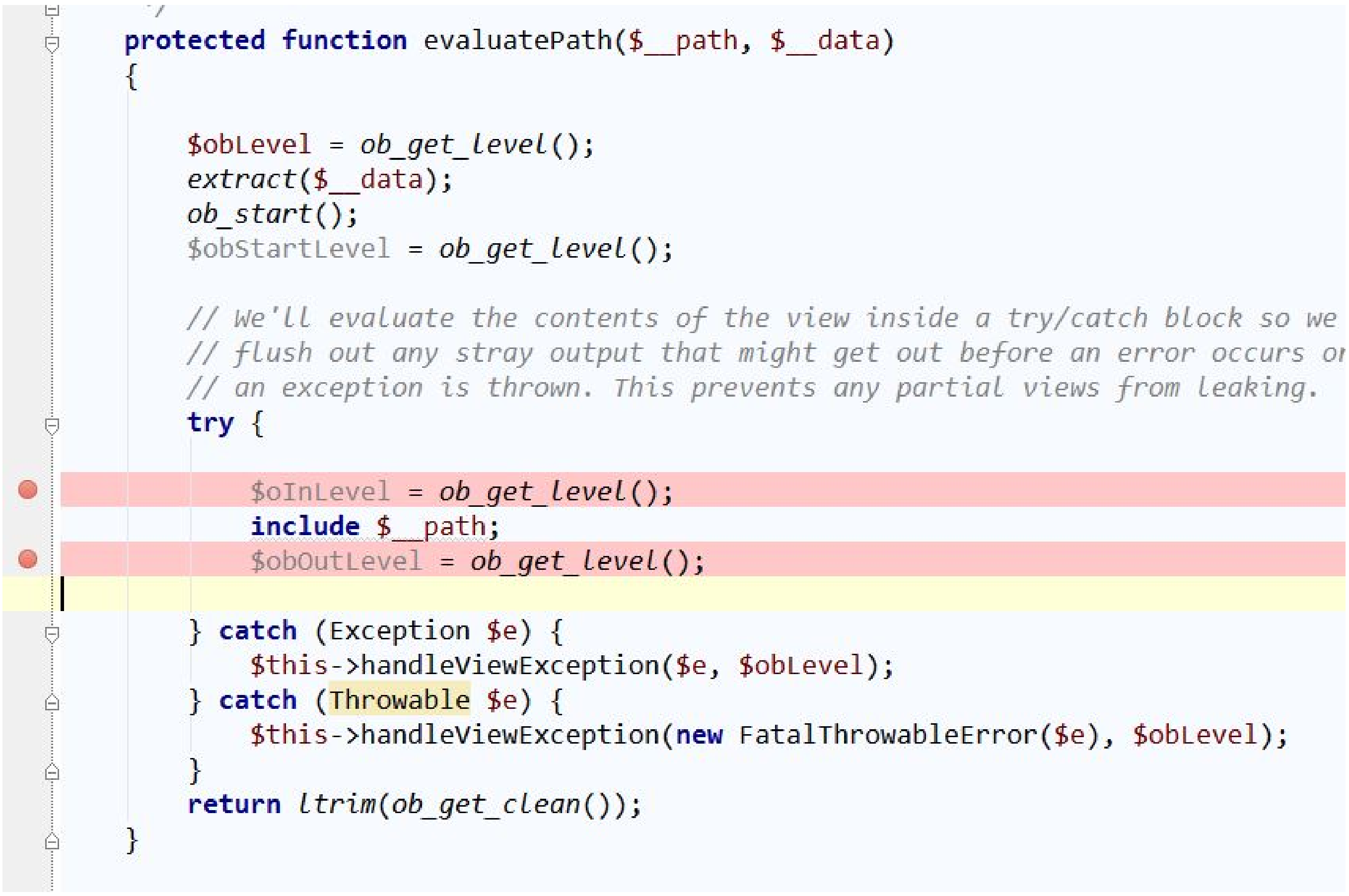


```
                    </div>
                </div>
            </div>
        </div>
</div>
<?php $__env->stopSection(); ?>
<?php echo $__env->make('_layouts.default', array_except(get_defined_vars(), array('__data', '__path')))->render(); ?>
```

It contains another view rendering and this process will go agian. It is a iterated process. In Laravel we can extend parent view from child view. So the process is that we render child view first, then the parent view, and return HTML code in last step of render.

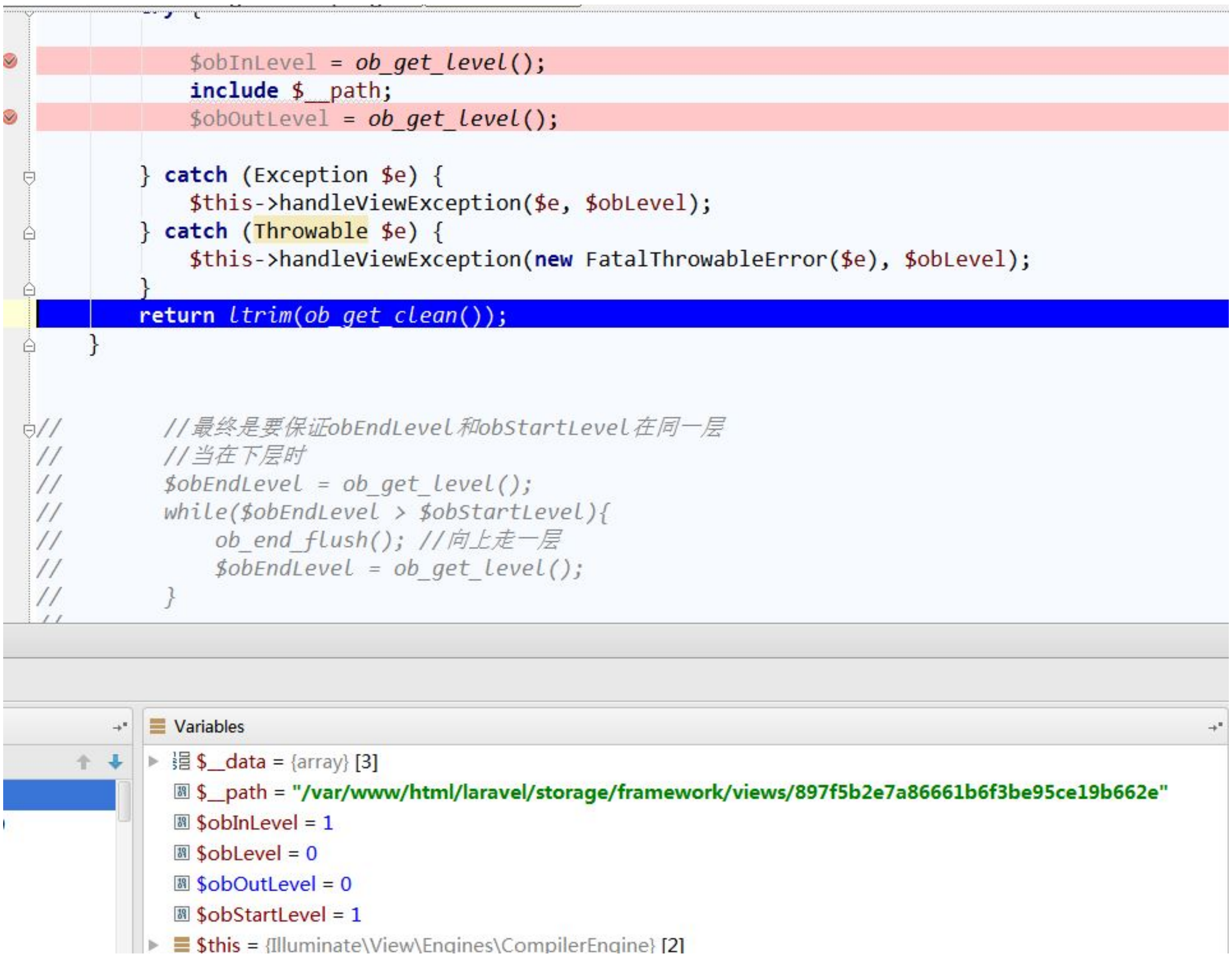
**How can we return HTML code string in the iterated process? **

From the function evaluatePath the answer is output*buffering, so the ob*start, ob*getclean, obget*level functions. And these functions are evil we know that. So I check if these functions have been correctedly executed. So I add two lines to check.



```
    protected function evaluatePath($__path, $__data)
    {

        $obLevel = ob_get_level();
        extract($__data);
        ob_start();
        $obStartLevel = ob_get_level();

        // We'll evaluate the contents of the view inside a try/catch block so we
        // flush out any stray output that might get out before an error occurs o
        // an exception is thrown. This prevents any partial views from leaking.
        try {

            $oInLevel = ob_get_level();
            include $__path;
            $obOutLevel = ob_get_level();

        } catch (Exception $e) {
            $this->handleViewException($e, $obLevel);
        } catch (Throwable $e) {
            $this->handleViewException(new FatalThrowableError($e), $obLevel);
        }
        return ltrim(ob_get_clean());
    }
```

The first level of iteration:

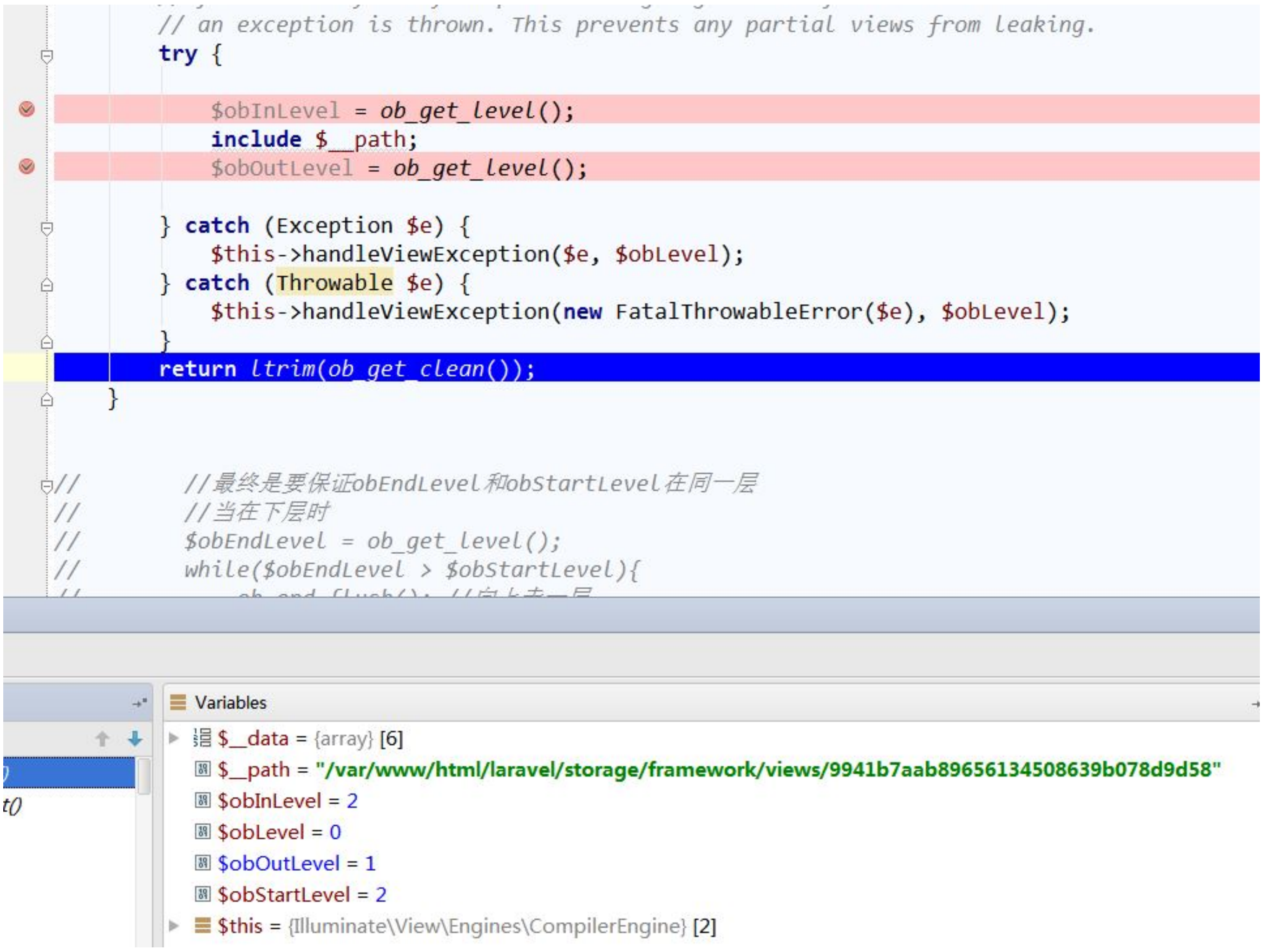


```
            $obInLevel = ob_get_level();
            include $__path;
            $obOutLevel = ob_get_level();

        } catch (Exception $e) {
            $this->handleViewException($e, $obLevel);
        } catch (Throwable $e) {
            $this->handleViewException(new FatalThrowableError($e), $obLevel);
        }
        return ltrim(ob_get_clean());
    }


//          //最终是要保证obEndLevel和obStartLevel在同一层
//          //当在下层时
//          $obEndLevel = ob_get_level();
//          while($obEndLevel > $obStartLevel){
//              ob_end_flush(); //向上走一层
//              $obEndLevel = ob_get_level();
//          }
```

Variables

```
▶ $__data = {array} [3]
  $__path = "/var/www/html/laravel/storage/framework/views/897f5b2e7a86661b6f3be95ce19b662e"
  $obInLevel = 1
  $obLevel = 0
  $obOutLevel = 0
  $obStartLevel = 1
▶ $this = {Illuminate\View\Engines\CompilerEngine} [2]
```

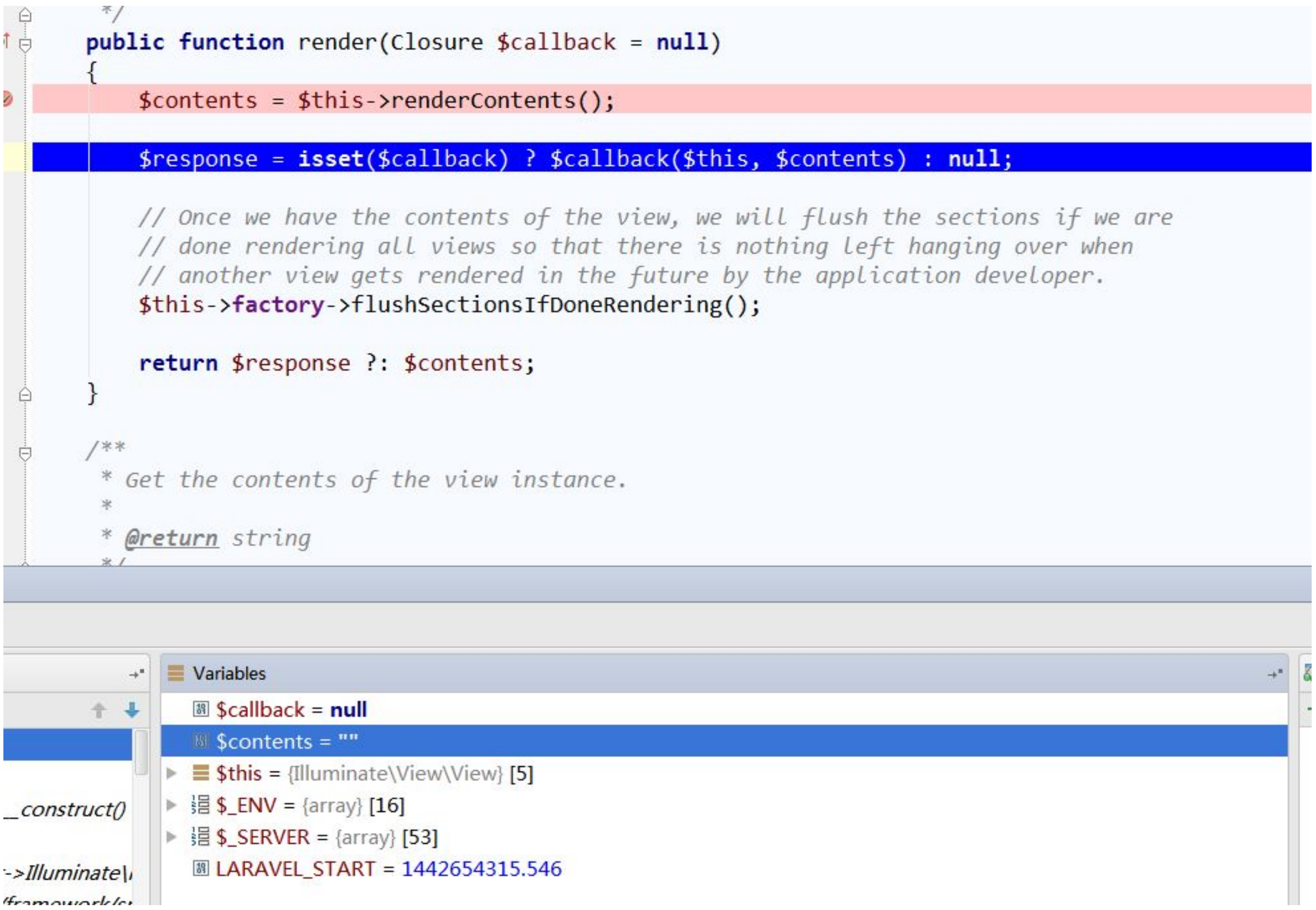
The second level of iteration:



```
        // an exception is thrown. This prevents any partial views from leaking.
        try {

            $obInLevel = ob_get_level();
            include $__path;
            $obOutLevel = ob_get_level();

        } catch (Exception $e) {
            $this->handleViewException($e, $obLevel);
        } catch (Throwable $e) {
            $this->handleViewException(new FatalThrowableError($e), $obLevel);
        }
        return ltrim(ob_get_clean());
    }

//          //最终是要保证obEndLevel和obStartLevel在同一层
//          //当在下层时
//          $obEndLevel = ob_get_level();
//          while($obEndLevel > $obStartLevel){
```

```
Variables
▶ $__data = {array} [6]
  $__path = "/var/www/html/laravel/storage/framework/views/9941b7aab89656134508639b078d9d58"
  $obInLevel = 2
  $obLevel = 0
  $obOutLevel = 1
  $obStartLevel = 2
▶ $this = {Illuminate\View\Engines\CompilerEngine} [2]
```

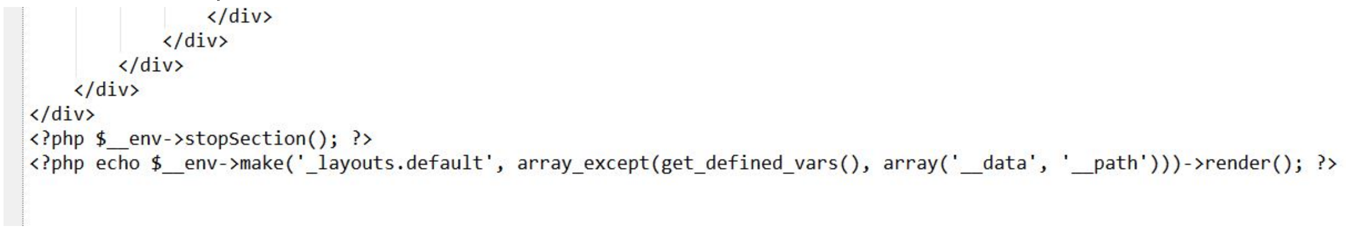
The result of iteration:



```
     */
    public function render(Closure $callback = null)
    {
        $contents = $this->renderContents();

        $response = isset($callback) ? $callback($this, $contents) : null;

        // Once we have the contents of the view, we will flush the sections if we are
        // done rendering all views so that there is nothing left hanging over when
        // another view gets rendered in the future by the application developer.
        $this->factory->flushSectionsIfDoneRendering();

        return $response ?: $contents;
    }

    /**
     * Get the contents of the view instance.
     *
     * @return string
```

```
Variables
  $callback = null
  $contents = ""
▶ $this = {Illuminate\View\View} [5]
▶ $_ENV = {array} [16]
▶ $_SERVER = {array} [53]
  LARAVEL_START = 1442654315.546
```

**We can see the $contents of result returned is empty. And the output*buffering level is different before "include $*_path" and after "include $*path". In the first level of iteration, the $obOutLevel is 0 before the code "return ltrim(obget*clean())" is executed. ** I remember there is a <?php echo ... ?> in the child compiled view file.



```
                </div>
            </div>
        </div>
    </div>
</div>
<?php $__env->stopSection(); ?>
<?php echo $__env->make('_layouts.default', array_except(get_defined_vars(), array('__data', '__path')))->render(); ?>
```

So it occurs to me this "echo" code in the child compiled view file is executed on the Output Buffering Level 0. The answer is clearly that content of echo in output buffering 0 is directly sent to browser. So There is nothing in the output buffering. If the contents of echo- ... -is sent to browser precedingly, How can the headers be sent back to broswer in the following steps of sending response ? That is why "the Headers have already been sent" error has gotten happened and the site shows fine. Then it triggers a series of things, cookie not sent, token mismatched.

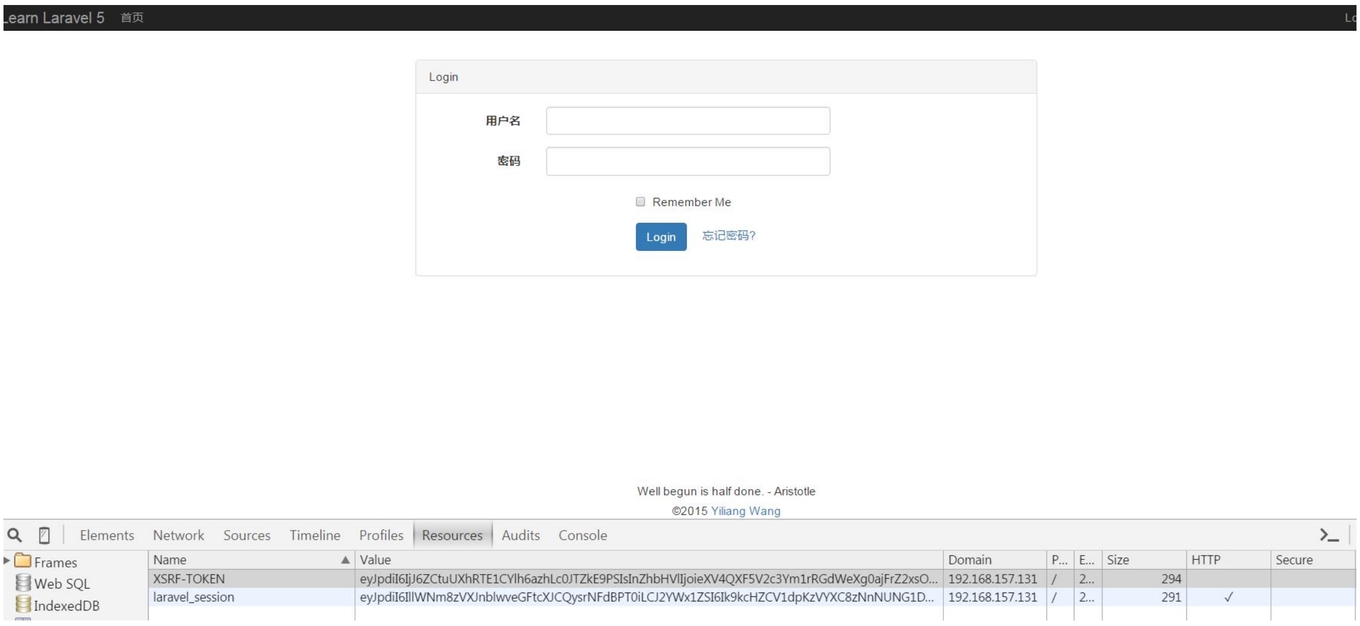
**And God knows why it jumps out OB level after executed include $__path, i.e. the compiled view file. I have observed this phenomenon for some times. This thing doesn't always happen related to different views. Like others this thing is random some times, So is the Token Mismatch Exception. I will find that later. **

5. My Solution

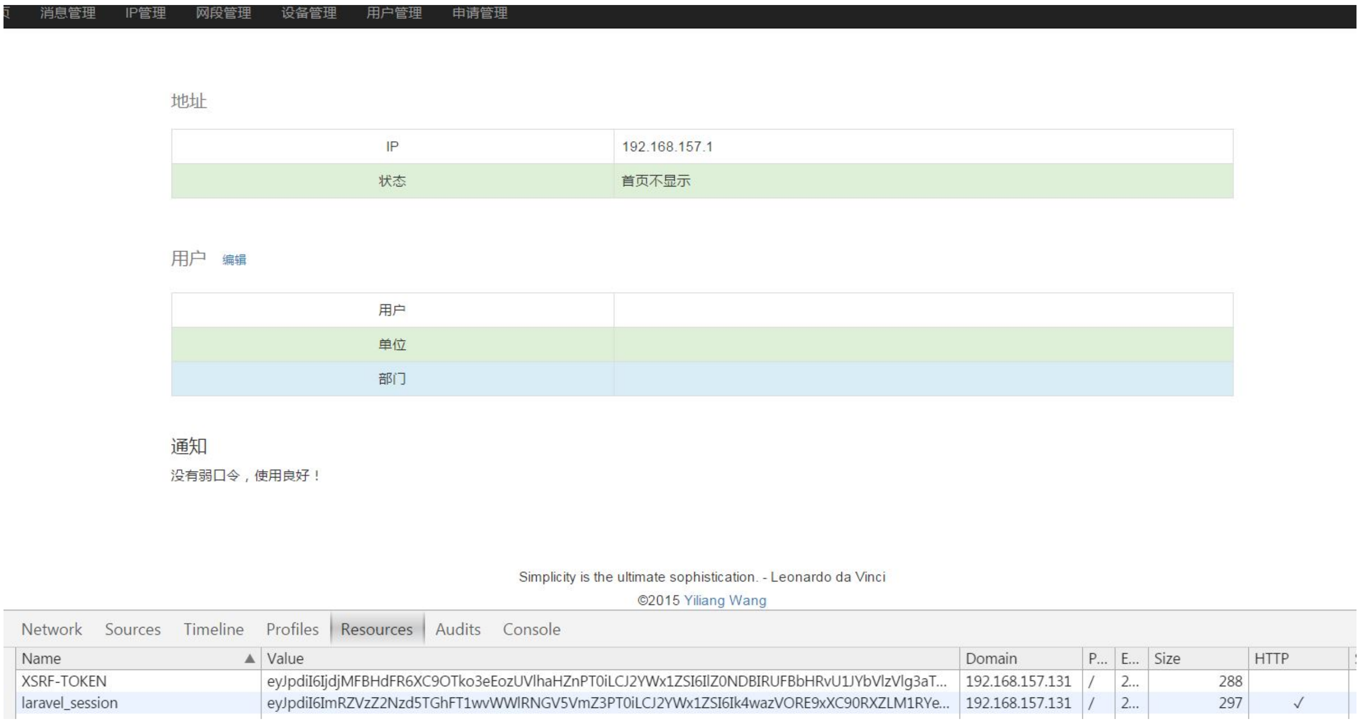
The solution is simple. That is to maintain it is in the same output buffering level after excuted "include $__path". So I change the evaluatePath function in file "Illuminate\View\Engines\PhpEngine.php". Here is my solution Code:

```
protected function evaluatePath($__path, $__data)
{

    $obLevel = ob_get_level();
    extract($__data);
    ob_start();
    $obStartLevel = ob_get_level();

    // We'll evaluate the contents of the view inside a try/catch block so we can
    // flush out any stray output that might get out before an error occurs or
    // an exception is thrown. This prevents any partial views from leaking.
    try {

        include $__path;

    } catch (Exception $e) {
        $this->handleViewException($e, $obLevel);
    } catch (Throwable $e) {
        $this->handleViewException(new FatalThrowableError($e), $obLevel);
    }

    //最终是要保证obEndLevel和obStartLevel在同一层
    //当在下层时
    $obEndLevel = ob_get_level();
    while($obEndLevel > $obStartLevel){
        ob_end_flush(); //向上走一层
        $obEndLevel = ob_get_level();
    }
    $myContent = ltrim(ob_get_contents()); //内容已经获得
    //当在上层时
    while($obEndLevel < $obStartLevel){
        ob_clean();  //只是清空内容，不能销毁上面的层次
        if(!ob_start())  break; //向下走一层
        $obEndLevel = ob_get_level();
    }
    //最后清掉本层
    if($obEndLevel === $obStartLevel) ob_end_clean();
    return $myContent;
    //return ltrim(ob_get_clean());
}
```

Then It works well. The token mismatch exception doesn't appear anymore.



Name	Value	Domain	P...	E...	Size	HTTP	Secure
XSRF-TOKEN	eyJpdiI6IjJ6ZCtuUXhRTE1CYlh6azhLc0JTZkE9PSIsInZhbHVlIjoieXV4QXF5V2c3Ym1rRGdWeXg0ajFrZ2xsO...	192.168.157.131	/	2...	294		
laravel_session	eyJpdiI6IllWNm8zVXJnblwveGFtcXJCQysrNFdBPT0iLCJ2YWx1ZSI6Ik9kcHZCV1dpKzVYXC8zNnNUNG1D...	192.168.157.131	/	2...	291	✓	



地址

IP	192.168.157.1
状态	首页不显示

用户 编辑

用户	
单位	
部门	

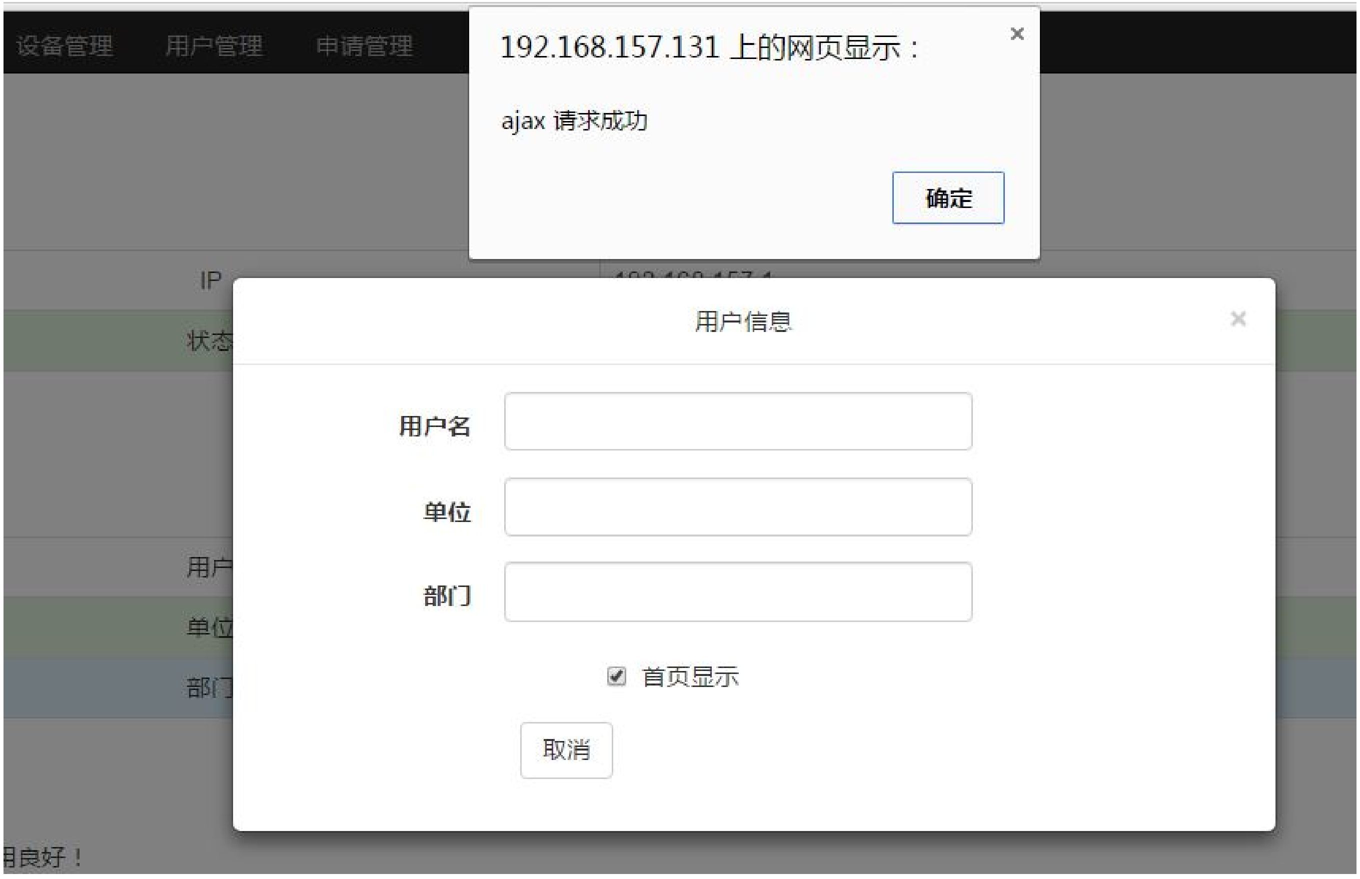
通知

没有弱口令，使用良好！

Simplicity is the ultimate sophistication. - Leonardo da Vinci

©2015 Yiliang Wang

Name	Value	Domain	P...	E...	Size	HTTP
XSRF-TOKEN	eyJpdiI6IjdjMFBHdFR6XC9OTko3eEozUVlhaHZnPT0iLCJ2YWx1ZSI6IlZ0NDBIRUFBbHRvU1JYbVlzVlg3aT...	192.168.157.131	/	2...	288	
laravel_session	eyJpdiI6ImRZVzZ2Nzd5TGhFT1wvWWlRNGV5VmZ3PT0iLCJ2YWx1ZSI6Ik4wazVORE9xXC90RXZLM1RYe...	192.168.157.131	/	2...	297	✓



192.168.157.131 上的网页显示：

ajax 请求成功

用户信息

用户名	
单位	
部门	

☑ 首页显示

6. The Author

```
Yiliang Wang, China. Email:wangyiliang206@163.com
```