

บทที่ 8

การใช้งาน CAPTCHA และ File Helper

หลังจากที่เราได้เรียนรู้การใช้งาน HTML, Form และ URL Helper มาแล้วในบทที่ผ่านมา สำหรับในบทนี้เป็นการใช้งาน CAPTCHA และ File Helper ซึ่งเป็นอีกตัวช่วยที่จะทำให้เราสามารถสร้างเว็บไซต์ได้ง่ายขึ้น โดย CAPTCHA เป็นเครื่องมือช่วยสร้างชุดอักษรหรือตัวเลขเพื่อใช้ในการตรวจสอบค่าก่อนที่จะทำขั้นตอนอื่นๆ ของโปรแกรม เช่น คำนวณโหลดหรือล็อกอิน เป็นต้น ซึ่งตัวอย่างที่นิยมใช้กันคือ reCAPTCHA (<http://www.google.com/recaptcha>)

ส่วน Helper อีกตัวคือ File Helper จะช่วยให้เราสามารถจัดการกับไฟล์ได้ง่ายขึ้น ไม่ว่าจะเป็นการสร้างไฟล์ การลบไฟล์ การเขียนไฟล์ หรือการกำหนดสิทธิ์ต่างๆ ของไฟล์ ซึ่งผู้อ่านจะได้ศึกษารายละเอียดของ Helper ในบทนี้กัน



การใช้งาน CAPTCHA Helper

CAPTCHA Helper เป็น Helper ที่ช่วยให้เราสามารถสร้างชุดอักษรหรือตัวเลข เพื่อให้ผู้ใช้พิมพ์เพื่อตรวจสอบความถูกต้องของตัวอักษรหรือตัวเลขก่อนที่จะทำการอย่างใดอย่างหนึ่ง เช่น ล็อกอิน ดาวน์โหลด หรือโพสต์ข้อความ ทั้งนี้เพื่อเป็นการป้องกันการทำ Spam โดย CAPTCHA ย่อมาจาก “Completely Automated Public Turing test to tell Computers and Humans Apart” สามารถศึกษาข้อมูลเพิ่มเติมได้ที่ <http://th.wikipedia.org/wiki/CAPTCHA>

การทำ CAPTCHA นั้น CodeIgniter ได้แก้ไขจากรุ่นก่อนที่ทำไว้เป็น Plugin ให้มาเป็น Helper แทน การจะใช้งาน CAPTCHA ได้ให้ทำการเปิดส่วนเสริม (Extension) GD ของ PHP ก่อนจึงจะสามารถใช้งานได้ ด้วยการแก้ไขไฟล์ php.ini โดยแก้ไขบรรทัด ดังนี้

```
extension = php_gd2.dll
```

จากนั้นรีสตาร์ทโปรแกรม Apache ใหม่ แล้วตรวจสอบว่า GD ทำงานหรือยัง โดยไปที่ Address bar ของเบราว์เซอร์ พิมพ์ <http://localhost/phpinfo.php> จะต้องปรากฏรายละเอียดของ GD ดังรูปที่ 8-1

gd

GD Support	enabled
GD Version	bundled (2.0.34 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
FreeType Version	2.1.9
T1Lib Support	enabled
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled
XBM Support	enabled

**ข้อมูลส่วนนี้น่าจะไม่ขัด
พิมพ์แล้วทำเป็นรูปแบบ
ตารางใหม่น่าจะดีกว่า**

รูปที่ 8-1

การเรียนรู้ CAPTCHA Helper

การที่เราจะใช้งาน CAPTCHA Helper ได้นั้น ให้ทำการโหลด Helper มาก่อนด้วยคำสั่ง `$this->load->helper()` โดยทำในส่วนของฟังก์ชัน `__construct()` ใน Controller ดังนี้

```
function __construct()
{
    parent::__construct();
    $this->load->helper(array('captcha', 'url')); //โหลด captcha และ url helper
    $this->load->library('session'); //โหลด Session library
}
```

จากคำสั่งข้างบนเราจะต้องโหลด Helper เพิ่มอีกตัวคือ url เนื่องจากเราจะมีการใช้งานฟังก์ชันใน URL Helper ด้วย และสุดท้ายคือต้องโหลดไลบรารี Session ทั้งนี้เพราะเราจะมีการใช้ตัวแปร Session ในการกำหนดระยะเวลาในการใช้งานรูปภาพด้วย

หรือจะทำการโหลดอัตโนมัติทุกครั้งก็ได้ เพียงแก้ไขไฟล์ `autoload.php` (`application/config/`) เพื่อให้โหลด Helper อัตโนมัติ โดยแก้ไขบรรทัด `$autoload['helper']` ดังตัวอย่าง

```
$autoload['helper'] = array('captcha', 'url');
$autoload['libraries'] = array('session');
```



จากนั้นเราต้องทำการกำหนดค่า Security key ให้กับ Session ด้วย โดยการแก้ไขไฟล์ `application/config/config.php` ให้เป็นค่าดังนี้

```
$config['encryption_key'] = 'XjmREljsdfEsjlsdlf';
```

โดยเราสามารถใส่คีย์อะไรเข้าไปก็ได้เพื่อให้โปรแกรมนำไปเข้ารหัสกับตัวแปร Session ต่อไป

หลักการทํางาน CAPTCHA Helper

หลังจากที่เราได้โหลด Library และ Helper สำหรับใช้งาน CAPTCHA แล้ว ในการใช้งาน CAPTCHA นั้นเราจะทำกันในส่วนของ Controller ซึ่งในขั้นตอนแรกเพื่อให้ผู้อ่านได้เข้าใจหลักการทํางาน CAPTCHA Helper ของ CodeIgniter เราจะศึกษารูปแบบการทํางานแบบง่าย ๆ ก่อน จากนั้นจึงจะไปพูดถึงเรื่องของการ Validation ต่อไป

ขั้นตอนแรกให้เราสร้างฟังก์ชันใน Controller ชื่อ captcha() ขึ้นมา โดยมีรายละเอียดของโค้ด ดังนี้  

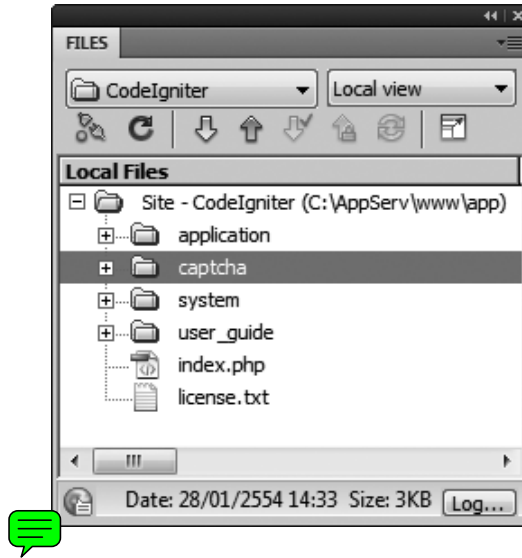
```
function captcha()
{
    $attr = array(
        'word' => '1435mL',
        'img_path' => './captcha/',
        'img_url' => base_url() . 'captcha/',
        'img_width' => 150,
        'img_height' => 30,
        'expiration' => 7200
    );
    $captcha = create_captcha($attr);
    echo $captcha['image'];
}
```

รายละเอียดของค่าต่าง ๆ ที่กำหนด

word เป็นการกำหนดข้อความที่ต้องการให้แสดงในภาพ หากไม่กำหนดโปรแกรมจะ ~~ทำ~~สุ่ม (Random) ให้อัตโนมัติ

img_path กำหนดที่อยู่ที่ใช้สำหรับเก็บภาพหลังจากสร้างเสร็จ ซึ่งจากตัวอย่างเรา

ต้องสร้างโฟลเดอร์ captcha ใหม่ในโฟลเดอร์หลักของเว็บไซต์ ตัวอย่างคือ C:\AppServ\www\app\captcha โดยโฟลเดอร์ของ CodeIgniter ผมเปลี่ยนจาก CodeIgniter เป็น app เพื่อให้ง่ายต่อการจดจำ โครงสร้างโฟลเดอร์ดังรูปที่ 8-2



รูปที่ 8-2

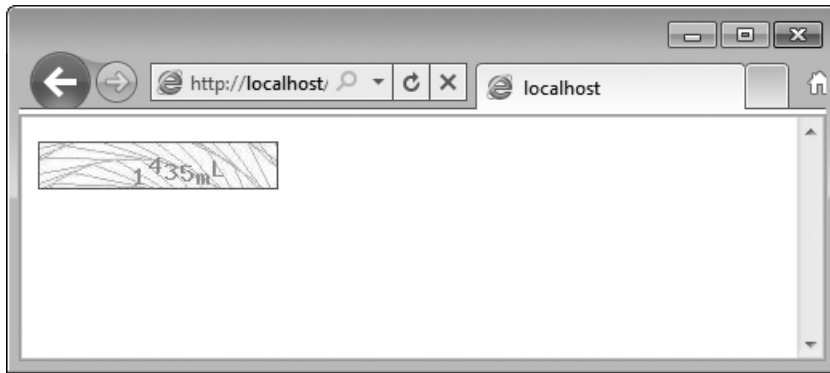
`img_url` เป็นที่อยู่ของรูปภาพที่ถูกสร้างขึ้นโดยใช้แท็ก `` ซึ่งเราจะใช้ฟังก์ชัน `base_url()` ใน URL Helper เพื่อสร้างลิงก์ให้

`img_width` เป็นการกำหนดค่าความกว้างของรูปภาพที่จะสร้าง

`img_height` กำหนดค่าความสูงของรูปภาพ

`expiration` กำหนดค่าเวลาที่ใช้สำหรับ Session กำหนดเป็นวินาที ซึ่งจะเป็นการกำหนดระยะเวลาที่ภาพจะถูกลบทิ้ง ค่าปกติคือ 7200 วินาที หรือ 2 ชั่วโมง ($7200 = 2 \times 60 \times 60$)

จากตัวอย่างของฟังก์ชัน `captcha()` เมื่อรันผ่านเบราว์เซอร์จะได้ ดังนี้



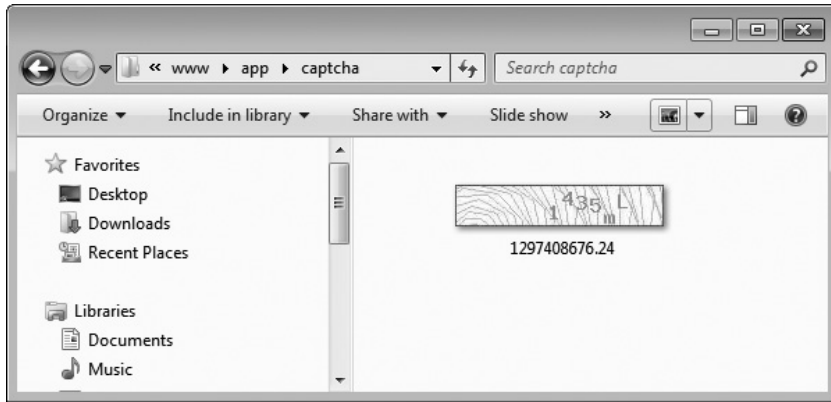
รูปที่ 8-3

ค่าที่ได้คืนกลับมาจากโปรแกรม คือ

```

Array
(
    [word] => 1435mL
    [time] => 1297408676.24
    [image] => 
)
  
```

เมื่อกลับไปดูที่ไฟล์เดอร์ C:\AppServ\www\app\captcha จะปรากฏรูปภาพที่สร้าง ขึ้น ดังรูปที่ 8-4



รูปที่ 8-4

จากคำสั่งในฟังก์ชัน `captcha()` เราจะมีกำหนดค่าต่าง ๆ เพื่อใช้สำหรับการสร้าง CAPTCHA โดยใช้ฟังก์ชัน `create_captcha()` เพื่อทำการสร้าง CAPTCHA สำหรับรูปแบบการใช้งานฟังก์ชัน `create_captcha()` มีดังนี้

```
create_captcha($data = '', $img_path = '', $img_url = '', $font_path = '')
```

โดยที่ `$data` เป็นการกำหนดคุณสมบัติต่าง ๆ ให้กับ CAPTCHA ซึ่งสามารถกำหนดเป็นตัวแปรแบบอาร์เรย์ได้ ค่าที่สามารถกำหนดได้ คือ `word`, `img_path`, `img_url`, `img_width`, `img_height`, `font_path` และ `expiration` ค่าเริ่มต้น คือ

```
$defaults = array(
    'word' => '', 'img_path' => '', 'img_url' => '', 'img_width' => '150',
    'img_height' => '30', 'font_path' => '', 'expiration' => 7200
)
```

**\$img_path**

กำหนดที่อยู่สำหรับเก็บรูปภาพที่ได้จากการสร้าง CAPTCHA

\$img_url

กำหนด url ที่ใช้เรียกรูปภาพที่สร้างเสร็จแล้ว

\$font_path

กำหนดที่พาทที่อยู่ของฟอนต์

ค่าที่ได้จากฟังก์ชัน create_captcha() จะคืนมาในรูปแบบ ดังนี้

Array



(

[word]

=> ตัวอักษรหรือตัวเลขที่ต้องการทำ CAPTCHA

[time]

=> เวลาในรูปแบบของ timestamp ซึ่งใช้สำหรับตั้งชื่อรูปภาพ

[image]

=> ที่อยู่ของรูปภาพโดยใช้แท็ก เป็นตัวกำหนดที่อยู่รูปภาพ

)

เราสามารถนำค่าที่ได้จากฟังก์ชัน create_captcha() ไปใช้งานเพื่อการตรวจสอบ (Validation) ต่อไป

ตัวอย่างการใช้งาน CAPTCHA

สำหรับตัวอย่างต่อไปนี้เป็นจะเป็นการใช้งาน CAPTCHA โดยจะใช้งานร่วมกับ Cookie เพื่อใช้ตรวจสอบค่าที่ส่งมาจากฟอร์ม ขั้นตอนแรกเราต้องทำการโหลด Library ชื่อ Session และ Helper ชื่อ form และ captcha ก่อน โดยเพิ่มโค้ดต่อไปนี้ในฟังก์ชัน __construct() ของ Controller ดังนี้



```
function __construct()
{
    parent::__construct();
    $this->load->helper(array('url', 'captcha', 'form'));
    $this->load->library('session');
}
```


ต่อไปให้เราสร้างฟังก์ชันใน Controller ขึ้นมา 2 ฟังก์ชัน โดยฟังก์ชันแรกชื่อ login() ส่วนฟังก์ชันที่สองชื่อ dologin() โดยมีโค้ดดังนี้

```
function login()
{
    $attr = array(
        'img_path'      => './captcha/',
        'img_url'       => base_url() . 'captcha/',
        'img_width'     => 150,
        'img_height'    => 30
    );
    $captcha = create_captcha($attr);
    setcookie('org_word', $captcha['word'], time() + 3600);
    echo $captcha['image'];
    echo br(2);
    echo form_open('welcome/dologin');
    echo form_input('word');
    echo form_submit('act', 'ตรวจสอบ');
    echo form_close();
}

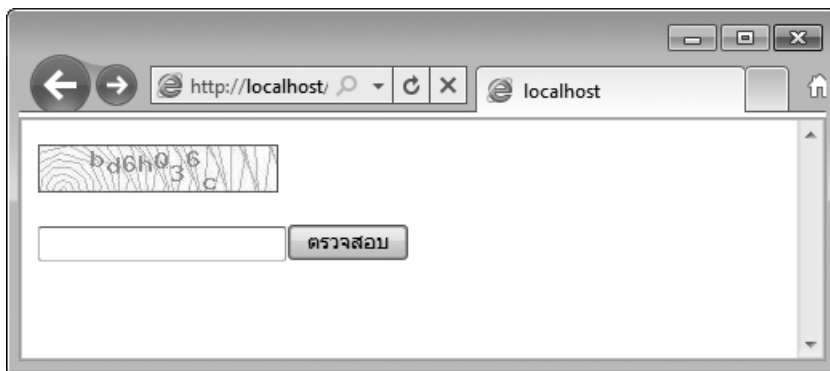
function dologin()
{
    if(! isset($_COOKIE['org_word']))
    {
        redirect('welcome/login');
    }else{
        $capt = $_COOKIE['org_word'];
    }
}
```



```
$word = $this->input->post('word');
if($word === $capt)
{
    echo 'ข้อมูลถูกต้อง';
}else{
    echo 'ข้อมูลไม่ถูกต้อง';
}
}
```

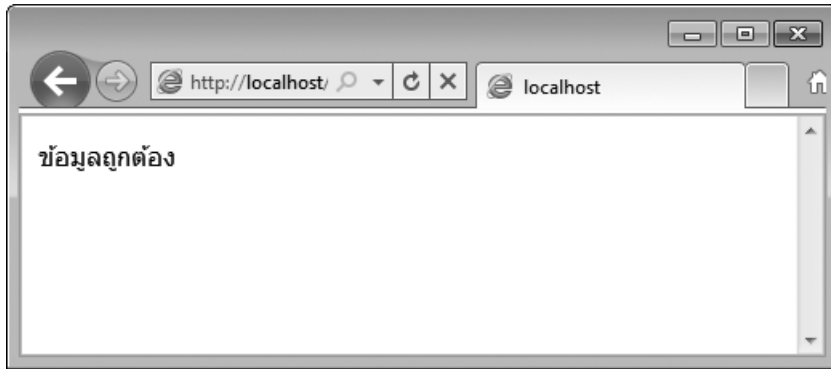


หลักจากนั้นทดสอบโปรแกรมโดยเปิดโปรแกรมเบราว์เซอร์ขึ้นมา แล้วพิมพ์ <http://localhost/app/index.php/welcome/login> (ตัวอย่างผมสร้างฟังก์ชัน login() และ dologin() ไว้ใน Controller ชื่อ Welcome) Address bar จะปรากฏหน้าจอ ดังรูป



รูปที่ 8-5

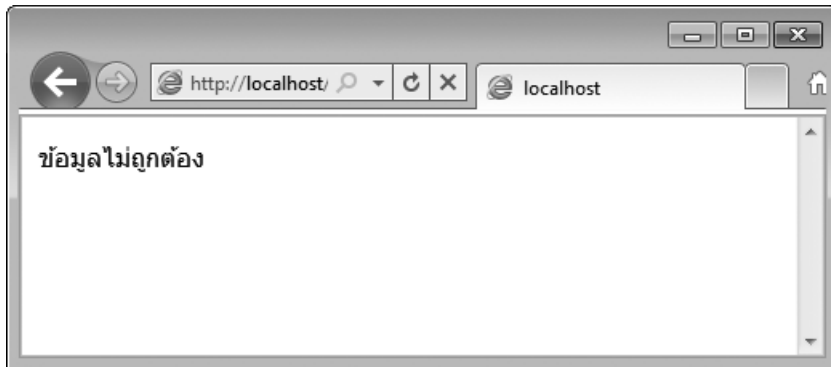
เมื่อผู้ใช้งานป้อนรหัสตามภาพที่ปรากฏลงในช่องว่าง (Input) แล้วคลิกที่ปุ่ม “ตรวจสอบ” หากพิมพ์ถูกต้องจะแสดงข้อความ ดังรูป



รูปที่ 8-6



แต่หากกรอกรหัสไม่ถูกต้องโปรแกรมจะแสดงข้อความ ดังรูป

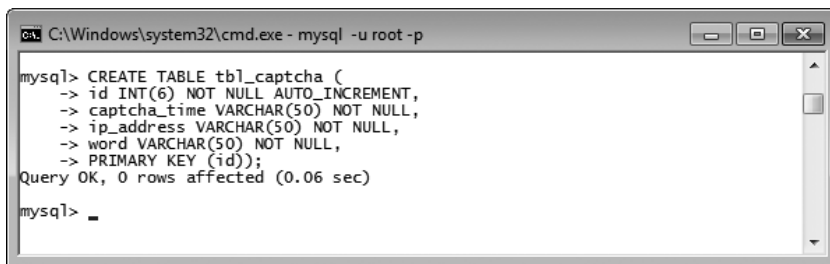


รูปที่ 8-7

จากโปรแกรมตัวอย่าง login() เป็นฟังก์ชันที่ใช้สำหรับสร้างภาพจาก CAPTCHA โดยใช้ฟังก์ชัน create_captcha() โดยให้สุ่มค่าตัวอักษรหรือตัวเลขอัตโนมัติจาก Helper จากนั้นนำค่าที่ได้เก็บไว้ใน Cookie ชื่อ org_word เมื่อผู้ใช้กรอกรหัสจากรูปภาพใน Input แล้วคลิกที่ปุ่ม “ตรวจสอบ” โปรแกรมจะทำการส่งค่าจากฟอร์มไปให้ฟังก์ชัน dologin() เพื่อตรวจสอบค่าที่ได้จาก Input ไปตรวจสอบกับ Cookie ชื่อ org_word หากตรงกันก็จะแสดงข้อความ “ข้อมูลถูกต้อง” แต่หากไม่ตรงกันโปรแกรมจะแสดงข้อความ “ข้อมูลไม่ถูกต้อง” ขึ้นมา

การใช้งาน CAPTCHA Helper กับฐานข้อมูล

สำหรับตัวอย่างต่อไปนี้เป็นการใช้ฐานข้อมูลเพื่อเก็บค่าตัวแปรของ word ที่ได้จากฟังก์ชัน create_captcha() จากนั้นเราจะใช้ Model เพื่อตรวจสอบค่าจากฟอร์มที่ผู้ใช้งานกรอกโดยในขั้นตอนแรกเราจะต้องสร้างฐานข้อมูลและตารางขึ้นมาก่อนเพื่อเก็บรายละเอียดของ CAPTCHA โดยใช้คำสั่ง SQL ดังนี้



```

C:\Windows\system32\cmd.exe - mysql -u root -p

mysql> CREATE TABLE tbl_captcha (
  -> id INT(6) NOT NULL AUTO_INCREMENT,
  -> captcha_time VARCHAR(50) NOT NULL,
  -> ip_address VARCHAR(50) NOT NULL,
  -> word VARCHAR(50) NOT NULL,
  -> PRIMARY KEY (id));
Query OK, 0 rows affected (0.06 sec)

mysql> _
  
```

รูปที่ 8-8

จากคำสั่งรูปที่ 8-8 เป็นการสร้างตารางขึ้นมาใหม่ ชื่อ tbl_captcha โดยมีรายละเอียดของฟิลด์ ดังนี้

id

เป็นรหัสอัตโนมัติเพื่อใช้เป็นคีย์สำหรับอ้างอิง

captcha_time สำหรับเก็บค่า timestamp ที่ใช้สร้างรูปภาพหรือก็คือชื่อรูปภาพนั่นเอง

ip_address ใช้สำหรับเก็บค่า IP ของเครื่องลูกข่าย

word เป็นค่าที่ใช้สำหรับการตรวจสอบซึ่งได้จากฟังก์ชัน `create_captcha()` ที่ส่งขึ้นมา

กำหนดค่าการเชื่อมต่อด้านข้อมูลด้วยการแก้ไขไฟล์ `application/config/database.php` ดังนี้

```
$db['default']['hostname'] = 'localhost'; //ที่อยู่ของ MySQL Server
$db['default']['username'] = 'root'; //ชื่อผู้ใช้งาน
$db['default']['password'] = '123456'; //รหัสผ่าน
$db['default']['database'] = 'users'; //ชื่อฐานข้อมูลที่เราสร้างตาราง tbl_captcha
```

สร้างคลาส Model ขึ้นมาใหม่ชื่อ `Captcha_model` (~~`application/models/captcha_model.php`~~) โดยเพิ่มโค้ดต่อไปนี้เข้าไป 

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Captcha_model extends CI_Model {

    function __construct()
    {
        parent::__construct();
    }

    #เพิ่มข้อมูล

    function _insert($time, $ip, $word){
        #ลบข้อมูลเก่าที่หมดอายุไปแล้ว

        $this->_delete($ip, $time);
```



```

#สร้างชุดคำสั่ง Active Record สำหรับเพิ่มข้อมูล
$result = $this->db->set('captcha_time', $time) //กำหนดเวลา
            ->set('ip_address', $ip) //กำหนด IP
            ->set('word', $word) //กำหนดชุดตัวอักษร
            ->insert('tbl_captcha');

    return $result;
}

#ตรวจสอบคำที่คีย์
function _check($ip, $word, $exp){
    $result = $this->db->where('captcha_time >', $exp)
                ->where('word', $word) //ตรวจสอบคำหรือ คีย์เวิร์ด
                ->where('ip_address', $ip) //ตรวจสอบ IP
                ->count_all_results('tbl_captcha');

    if($result > 0) //พบรายการ
    {
        return TRUE;
    }else{ //ไม่พบรายการ
        return FALSE;
    }
}

#ลบข้อมูลเดิมทั้งหมดอายุ
function _delete($ip, $exp)
{
    $this->db->where('ip_address', $ip)
                ->where('captcha_time <', $exp)
                ->delete('tbl_captcha');
}
}

```

ฟังก์ชัน `_insert($time, $ip, $word)` จะทำหน้าที่ในการเพิ่มข้อมูลเวลา (time), ไอพี (ip) และรหัส (word) ที่ได้จากการใช้ฟังก์ชัน `create_captcha()` ลงในตาราง `tbl_captcha`

ฟังก์ชัน `_check($ip, $word, $exp)` จะทำหน้าที่ตรวจสอบค่าที่มาจาก Controller ซึ่งเป็นค่าของรหัส (word), ไอพี (ip) และเวลา (time) ที่ผู้ใช้งานกรอกลงในช่อง (Input) ซึ่งถ้าข้อมูลตรงกับที่อยู่ในตาราง ฟังก์ชันนี้จะคืนค่า TRUE กลับไป แต่ถ้าไม่ถูกต้อง ฟังก์ชันก็จะคืนค่า FALSE กลับไปแทน

ฟังก์ชัน `_delete($ip, $exp)` ใช้สำหรับลบข้อมูลเดิมที่ผ่านการใช้งานมาแล้วหรือหมดอายุ (Expired) ออกจากตาราง `tbl_captcha` เพื่อไม่ให้ข้อมูลซ้ำซ้อนกัน

~~ให้สร้าง Controller ใหม่ ซึ่งตัวอย่างนี้ใช้ Controller ชื่อ Welcome (application/controllers/welcome.php) มีรายละเอียดของโค้ด ดังนี้~~

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');
```

```
class Welcome extends CI_Controller {
```

```
    function __construct()
```

```
{
```

```
        parent::__construct();
```

```
        $this->load->helper(array('url', 'captcha', 'form'));
```

```
        $this->load->library('session');
```

```
        $this->load->model('Captcha_model', 'Captcha');
```

```
}
```

```
    function login()
```

```
{
```

```
        $attr = array(
```

```
            'img_path' => './captcha/',
```

```
            'img_url' => base_url() .
```

```
            'captcha/',
```



```

        'img_width'      => 150,
        'img_height'     => 30
    );
    #create captcha
    $captcha      = create_captcha($attr);
    $time         = $captcha['time'];
    $ip           = $this->input->ip_address();
    $word         = $captcha['word'];
    #insert data
    $this->Captcha->_insert($time, $ip, $word);
    #create form
    echo $captcha['image'];
    echo br(2);
    echo form_open('welcome/dologin');
    echo form_input('word');
    echo form_submit('act', 'ตรวจสอบ');
    echo form_close();
}
function dologin()
{
    $act = $this->input->post('act');
    if(empty($act))
    {
        redirect('welcome/login');
    }
    else{
        $word = $this->input->post('word');
        $exp = time() - 7200;
    }
}

```




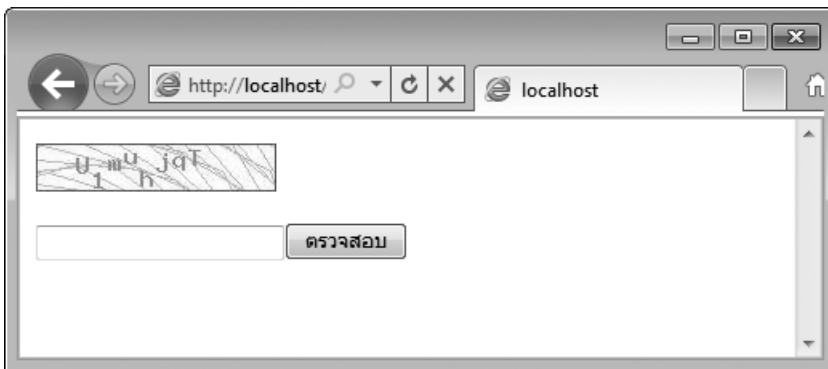


```
$ip = $this->input->ip_address();  
#check captcha word  
$result = $this->Captcha->_check($ip, $word, $exp);  
if($result)  
{  
    echo 'ข้อมูลถูกต้อง';  
}  
else{  
    echo 'ข้อมูลไม่ถูกต้อง';  
}  
}  
}
```

ฟังก์ชัน login() จะทำหน้าที่ในการสร้าง CAPTCHA ขึ้นมา แล้วส่งค่าให้ฟังก์ชัน _insert() ในคลาส Captcha_model ทำการเพิ่มข้อมูลลงในตารางเพื่อใช้สำหรับตรวจสอบอีกครั้ง และจะทำการสร้างฟอร์มสำหรับให้ผู้ใช้กรอกรหัส (word) เพื่อตรวจสอบกับค่าในตารางอีกที

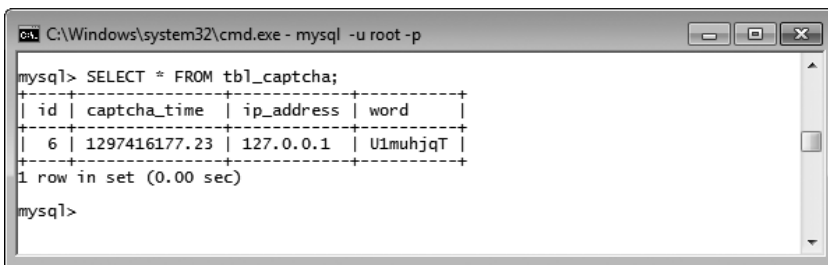
ฟังก์ชัน dologin() จะทำหน้าที่ในการตรวจสอบค่ารหัส (word) ที่ส่งมาจากฟอร์มกับฟังก์ชัน _check() ที่อยู่ในคลาส Captcha_model ซึ่งฟังก์ชัน _check() จะตรวจสอบค่าที่ส่งมากับค่าที่อยู่ในฐานข้อมูลโดยการอ้างอิงค่า ip_address และ word ในตาราง tbl_captcha

เมื่อเราสร้างไฟล์ต่าง ๆ เสร็จเรียบร้อยแล้ว ต่อมาเป็นการทดสอบการทำงานของโปรแกรมว่าสามารถทำงานได้หรือไม่ โดยเปิดโปรแกรมเบราว์เซอร์ แล้วพิมพ์ <http://localhost/app/index.php/welcome/login> จะ  ภูหน้าจอ ดังรูป



รูปที่ 8-9

เมื่อตรวจสอบดูข้อมูลที่อยู่ในตาราง จะพบว่ามีเรคอร์ดปรากฏขึ้นมาใหม่ ดังรูป



รูปที่ 8-10

ทุกครั้งที่มีการรีเฟรช หน้าจอโปรแกรมจะทำการลบรายการเก่าทิ้ง จากนั้นจึงสร้างรายการใหม่ของ CAPTCHA ขึ้นมาแทนโดยการอาศัยค่าไอฟีเป็นตัวอ้างอิง

ต่อไปให้กรอกรหัสจากรูปภาพลงในช่อง (input) แล้วคลิกที่ปุ่ม **“ตรวจสอบ”** โปรแกรมจะทำการส่งค่าจากฟอร์มไปให้ฟังก์ชัน `dologin()` เพื่อตรวจสอบค่าที่ได้จากช่อง (input) ไป

ตรวจสอบกับฟิลด์ word ที่อยู่ในตาราง tbl_captcha หากตรงกันก็จะแสดงข้อความ “**ข้อมูลถูกต้อง**” แต่หากไม่ตรงกัน โปรแกรมจะแสดงข้อความ “**ข้อมูลไม่ถูกต้อง**” ขึ้นมา

จากตัวอย่างการใช้งานที่ผ่านมาเป็นเพียงแค่การแสดงวิธีการใช้งานแบบง่าย ๆ ใน CAPTCHA Helper ของ CodeIgniter ทั้งนี้เพื่อให้ผู้อ่านสามารถนำไปประยุกต์ใช้งานจริงได้ โดยอาจจะใช้ตัวแปร Session เพื่อกำหนดค่า \$word ให้กับ CAPTCHA ก็ได้ แล้วทำการสุ่มค่าชุดตัวอักษรมากำหนดใน \$word หรืออาจเปลี่ยนรูปแบบตัวอักษรสำหรับทำภาพเพิ่มเติมให้ดูสวยงามหรือน่าสนใจกว่านี้ก็ได้

การใช้งาน File Helper

File Helper เป็นเครื่องมือที่จะช่วยให้เราสามารถจัดการกับไฟล์ได้ง่ายขึ้น ไม่ว่าจะเป็นการสร้างไฟล์ การอ่านไฟล์ การลบไฟล์ หรือดูข้อมูลของไฟล์ เราไม่จำเป็นต้องเขียนโดยใช้ฟังก์ชันของ PHP ให้ยุ่งยาก แต่เพียงแค่อ้างอิงฟังก์ชันที่อยู่ใน File Helper ก็สามารถจัดการกับไฟล์ได้เหมือนกับการเขียนโค้ด PHP ยาว ๆ

สำหรับในหัวข้อนี้เราจะได้เรียนรู้การใช้งานฟังก์ชันต่าง ๆ ของ File Helper พร้อมกับตัวอย่างการใช้งานเพื่อให้ผู้อ่านได้ศึกษาและทำความเข้าใจในการใช้งานได้ง่ายขึ้น

การเรียกใช้ File Helper

การที่เราจะใช้งาน Helper ได้นั้น ให้โหลด Helper มาก่อนด้วยคำสั่ง `$this->load->helper()` โดยทำในส่วนของฟังก์ชัน `__construct()` ใน Controller ดังนี้

```
function __construct()
{
    parent::__construct();
    $this->load->helper("file");
}
```

หรือจะทำการโหลดอัตโนมัติทุกครั้งก็ได้ เพียงแก้ไขไฟล์ `autoload.php` (`application/config/`) เพื่อให้โหลด Helper อัตโนมัติ โดยแก้ไขบรรทัด `$autoload['helper']` ดังตัวอย่าง

```
$autoload['helper'] = array('file');
```

ฟังก์ชันที่สามารถเรียกใช้งานได้ มีดังนี้

HTML Helper	ความหมาย	รูปแบบ
<code>read_file()</code>	สำหรับอ่านข้อมูลจากไฟล์	read_file (<i>file</i>)
<code>write_file()</code>	สำหรับเขียนข้อมูลลงไฟล์	write_file (<i>path, data, mode</i>)
<code>delete_files()</code>	สำหรับลบไฟล์โดยสามารถลบได้ทั้งโฟลเดอร์หรือเฉพาะไฟล์ก็ได้	delete_files (<i>path, del_dir, level</i>)
<code>get_filenames()</code>	สำหรับอ่านข้อมูลของไฟล์ที่อยู่ในโฟลเดอร์	get_filenames (<i>source_dir, include_path, recursion</i>)
<code>get_dir_file_info()</code>	สำหรับอ่านข้อมูลของไฟล์ที่อยู่ในโฟลเดอร์ไม่ว่าจะเป็น สิทธิ์การใช้ งาน ขนาดของไฟล์ วันที่สร้าง	get_dir_file_info (<i>source_dir, top_level_only, recursion</i>)
<code>get_file_info()</code>	แสดงรายละเอียดของไฟล์ที่กำหนด เช่น วันที่ สิทธิ์การใช้งาน ขนาด พาท	get_file_info (<i>file, return_value</i>)





HTML Helper	ความหมาย	รูปแบบ
<code>get_mime_by_extension()</code>	สำหรับแสดงข้อมูล mime type ของไฟล์โดยการตรวจสอบส่วนขยาย (Extension) ของไฟล์ (นามสกุลไฟล์หลังเครื่องหมาย '.')	<code>get_mime_by_extension(file)</code>
<code>symbolic_permissions()</code>	สำหรับแปลงข้อมูลสิทธิ์การใช้งาน (Permission) ของไฟล์ จากตัวเลขหรือเลขฐาน 8 ให้อยู่ในรูปแบบสัญลักษณ์ เช่น rw-r-x	<code>symbolic_permissions(permission)</code>
<code>octal_permissions()</code>	สำหรับการแปลงข้อมูลสิทธิ์การใช้งานให้อยู่ในรูปตัวเลขฐาน 8 เช่น 755, 777	<code>octal_permissions(permission)</code>

การอ่านข้อมูลจากไฟล์

การอ่านข้อมูลจากไฟล์เราจะใช้ฟังก์ชัน `read_file()` โดยมีรูปแบบการใช้งาน ดังนี้

```
read_file($file)
```

โดยที่ `$file` กำหนดชื่อของไฟล์ที่ต้องการอ่านข้อมูล

ตัวอย่างการใช้งานฟังก์ชัน `read_file()` ขึ้นแรกให้เราสร้างไฟล์ขึ้นมาใหม่ชื่อ `test.txt` โดยเก็บไว้ที่ `C:\Temp` และเพิ่มข้อความในไฟล์ ดังนี้

```
Hello world
```

จากนั้นแก้ไข Controller ชื่อ Welcome (application/controllers/welcome.php) :

จะสร้างขึ้นมาใหม่ โดยมีรายละเอียดของโค้ด ดังนี้

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

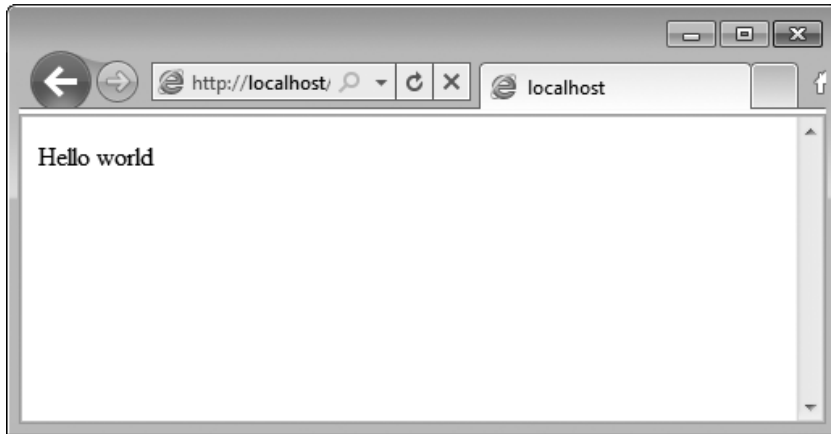
class Welcome extends CI_Controller {

    function __construct()
    {
        parent::__construct();
        $this->load->helper('file');
    }

    function file_detail()
    {
        $file = 'C:\\Temp\\test.txt';
        $data = read_file($file);
        if($data)
        {
            echo $data;
        }else{
            echo 'ไม่สามารถอ่านไฟล์ได้';
        }
    }

}
```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ http://localhost/app/index.php/welcome/file_detail จะปรากฏหน้าจอ ดังนี้



รูปที่ 8-11

การเขียนข้อมูลลงไฟล์

การเขียนข้อมูลลงไฟล์จะใช้ฟังก์ชัน `write_file()` โดยมีรูปแบบการใช้งาน ดังนี้

`write_file($path, $data, $mode = FOPEN_WRITE_CREATE_DESTRUCTIVE)`

โดยที่

- `$path` กำหนดพาทที่อยู่ของไฟล์ที่ต้องการเขียน
- `$data` กำหนดข้อมูลที่ต้องการจะเขียนลงในไฟล์
- `$mode` กำหนดโหมด (mode) ที่ใช้ในการเขียน ดูรายละเอียดได้ที่

`application/config/constant.php`

หมายเหตุ

โหมดที่ใช้ในการเขียนไฟล์สามารถกำหนดได้ ดังนี้

`rb` อ่านไฟล์อย่างเดียว

`r+b` อ่านและเขียนไฟล์ โดยข้อมูลเดิมจะถูกเขียนทับ

`wb` เขียนไฟล์ โดยเป็นการเขียนทับข้อมูลเดิม



w+b อ่านและเขียนไฟล์ โดยเป็นการเขียนทับข้อมูลเดิม หากไม่พบไฟล์ โปรแกรมจะสร้างขึ้นใหม่

ab เขียนไฟล์อย่างเดียว โดยเป็นการเขียนเพิ่มจากข้อมูลเดิม

a+b อ่านและเขียนไฟล์ โดยเป็นการเขียนเพิ่มจากข้อมูลเดิม

xb สร้างและเปิดไฟล์เพื่อเขียนอย่างเดียว ถ้ามีไฟล์เดิมอยู่โปรแกรมจะไม่สามารถเขียนได้ แต่ถ้าไม่พบไฟล์โปรแกรมจะสร้างไฟล์ขึ้นมาใหม่

x+b สร้างไฟล์ เพื่ออ่านและเขียน โดยจะเหมือนกับ x

ส่วนค่า b ที่ต่อท้ายจะเป็นการกำหนดรูปแบบการเปิดไฟล์ โดยตัวอย่างจะเป็นการเปิดหรือเขียนไฟล์แบบ Binary

ตัวอย่างการใช้งานฟังก์ชัน `write_file()` เราจะทำการเขียนข้อมูลลงในไฟล์ `test.txt` ที่อยู่ใน `C:\Temp` โดยให้สร้างฟังก์ชันขึ้นมาใหม่ ชื่อ `dowrite()` โดยมีรายละเอียดของโค้ด ดังนี้

```
function dowrite()
{
    $file = 'C:\\Temp\\test.txt';
    $data = 'My name is Satit Rianpit';
    $result = write_file($file, $data, 'r+b');
    if($result)
    {
        $data = read_file($file);
        if($data)
        {
            echo $data;
        }else{
            echo 'ไม่สามารถอ่านไฟล์ได้';
        }
    }
}
```




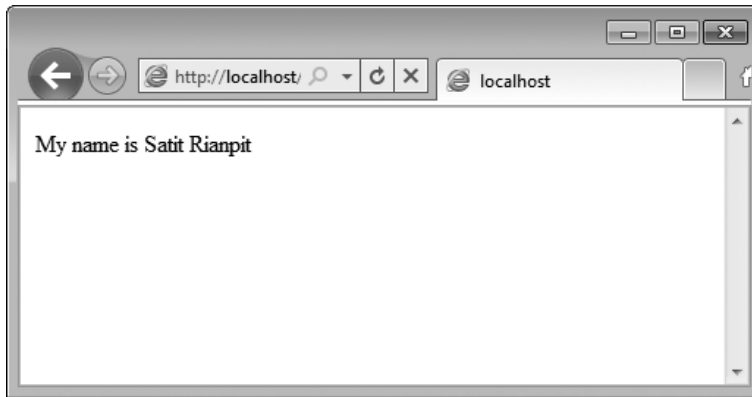


```

    }
    }else{
        echo 'ไม่สามารถเขียนไฟล์ได้';
    }
}

```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ <http://localhost/app/index.php/welcome/dowrite> จะ  กฎหน้าจอ ดังนี้



รูปที่ 8-12

การลบไฟล์


การลบไฟล์เราจะใช้ฟังก์ชัน `delete_file()` โดยมีรูปแบบการใช้งาน ดังนี้

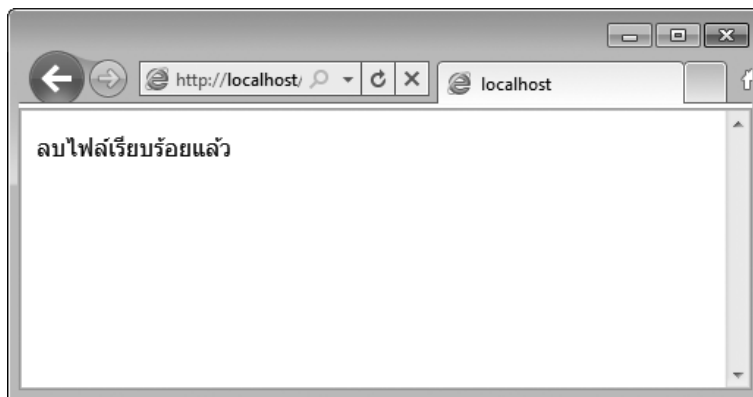
```
delete_files($path, $del_dir = FALSE, $level = 0)
```

โดยที่ `$path` กำหนดที่อยู่ของไฟล์หรือโฟลเดอร์ที่ต้องการลบ
`$del_dir` กำหนดให้มีการลบโฟลเดอร์ย่อย
`$level` ระดับของโฟลเดอร์ย่อยที่ต้องการจะลบ ค่าปกติคือ 0

ตัวอย่างการใช้งานฟังก์ชัน `delete_files()` เราจะทำการลบทุกไฟล์ที่อยู่ในโฟลเดอร์ `C:\Temp` โดยให้สร้างฟังก์ชันขึ้นมาใหม่ใน Controller ใช้ชื่อ `dodelete()` ซึ่งมีรายละเอียดของโค้ด ดังนี้

```
function dodelete()
{
    $file = 'C:\\Temp\\';
    $result = delete_files($file);
    if($result)
    {
        echo 'ลบไฟล์เรียบร้อยแล้ว';
    }else{
        echo 'ไม่สามารถลบไฟล์ได้';
    }
}
```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ <http://localhost/app/index.php/welcome/dodelete>  จากหน้าจอ ดังนี้



รูปที่ 8-13

การดูรายละเอียดของไฟล์

สำหรับในหัวข้อนี้เราจะเรียนรู้การใช้ฟังก์ชันต่างๆ ของ File Helper เพื่อใช้ในการอ่านข้อมูลของไฟล์หรือโฟลเดอร์ เช่น การแสดงชื่อไฟล์ในโฟลเดอร์ การแสดงคุณสมบัติต่างๆ ของไฟล์ การแสดงข้อมูลสิทธิหรือ Permission ของไฟล์ เป็นต้น ซึ่งฟังก์ชันการใช้งานต่างๆ มีดังนี้

• แสดงรายชื่อของไฟล์ในโฟลเดอร์

การแสดงรายชื่อของไฟล์ที่อยู่ในโฟลเดอร์ที่กำหนดเราจะใช้ฟังก์ชัน `get_filenames()` โดยมีรูปแบบการใช้งาน ดังนี้

```
get_filenames($source_dir, $include_path = FALSE, $_recursion = FALSE)
```

โดยที่	<i>\$source_dir</i>	ที่อยู่ของโฟลเดอร์ที่ต้องการจะแสดงรายชื่อของไฟล์
	<i>\$include_path</i>	กำหนดให้แสดงชื่อพารของไฟล์ ค่าปกติคือ FALSE
	<i>\$_recursion</i>	ให้มีการแสดงรายชื่อไฟล์ที่อยู่ในโฟลเดอร่อย่อยด้วย ค่าปกติคือ FALSE

ตัวอย่างการใช้งานฟังก์ชัน `get_filenames()` ให้เราสร้างฟังก์ชันใน Controller ขึ้นมาใหม่ชื่อ `dolist()` โดยมีรายละเอียดของโค้ด ดังนี้

```
function dolist()
{
    $path = 'C:\\Temp\\PCU\\';
    $files = get_filenames($path);
    if($files)
    {
        echo 'รายชื่อไฟล์ใน ' . $path . '<br />';
        foreach ($files as $f)
            echo $f . '<br />';
    }
}
```

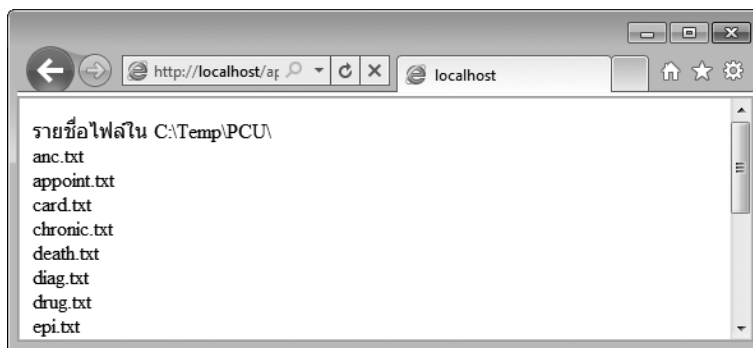


```

    }else{
        echo 'ไม่สามารถดูข้อมูลได้';
    }
}

```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ <http://localhost/app/index.php/welcome/dolist> จะปรากฏหน้าจอ ดังนี้



รูปที่ 8-14

• แสดงรายชื่อไฟล์พร้อมข้อมูลของไฟล์

การแสดงรายชื่อไฟล์พร้อมกับข้อมูลของไฟล์ที่อยู่ในโฟลเดอร์เราจะใช้ฟังก์ชัน `get_dir_file_info()` โดยมีรูปแบบการใช้งาน ดังนี้

```
get_dir_file_info($source_dir, $top_level_only = TRUE, $recursion = FALSE)
```

โดยที่ `$source_dir` กำหนดพารามิเตอร์ของโฟลเดอร์ที่ต้องการ

`$top_level_only` กำหนดค่าการแสดงผลเฉพาะไฟล์ในโฟลเดอร์หลัก ค่าปกติ คือ TRUE

`$recursion` ตรวจสอบเครื่องหมาย / ของ `$source_dir` ค่าปกติคือ FALSE

ค่าที่ส่งคืนกลับมาจากฟังก์ชัน `get_dir_file_info()` จะอยู่ในรูปแบบของอาร์เรย์ โดยมีรายละเอียดดังนี้

```
Array (
    'name' => ชื่อของไฟล์,
    'server_path' => ชื่อพารของไฟล์,
    'size' => ขนาดของไฟล์หน่วยเป็นไบต์ (Byte),
    'date' => วันที่มีการปรับปรุงแก้ไขล่าสุดอยู่ในรูปแบบของ time-
stamp,
    'relative_path' => ชื่อพารหลักของไฟล์
)
```

ตัวอย่างการใช้งานฟังก์ชัน `get_dir_file_info()` ให้เราสร้างฟังก์ชันใน Controller ขึ้นมาใหม่ชื่อ `dodirinfo()` โดยมีรายละเอียดของโค้ด ดังนี้

```
function dodirinfo()
{
    $path = 'C:\\Temp\\PCU\\';
    $files = get_dir_file_info($path);
    if($files)
    {
        echo 'รายชื่อไฟล์ใน ' . $path . '<br />';
        echo '
        <table width="100%" border="1">
            <tr>
                <td>ชื่อไฟล์</td>
                <td>พาร</td>
                <td>ขนาด (Byte)</td>
```



```


<td>วันที่</td>

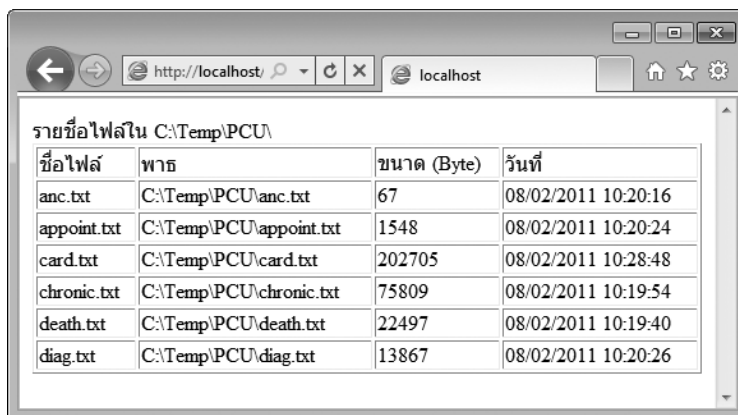
</tr>

';

foreach ($files as $f)
{
    echo '<tr><td>' . $f['name'] . '</td>';
    echo '<td>' . $f['server_path'] . '</td>';
    echo '<td>' . $f['size'] . '</td>';
    echo '<td>' . date('d/m/Y H:i:s' , $f['date']) . '</td></tr>';
}
echo '</table>';
}else{
    echo 'ไม่สามารถดูข้อมูลได้';
}
}

```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ <http://localhost/app/index.php/welcome/dodirinfo> จากนั้นกดปุ่ม  หน้าจอ ดังนี้



รายชื่อไฟล์ใน C:\Temp\PCU\

ชื่อไฟล์	พาท	ขนาด (Byte)	วันที่
anc.txt	C:\Temp\PCU\anc.txt	67	08/02/2011 10:20:16
appoint.txt	C:\Temp\PCU\appoint.txt	1548	08/02/2011 10:20:24
card.txt	C:\Temp\PCU\card.txt	202705	08/02/2011 10:28:48
chronic.txt	C:\Temp\PCU\chronic.txt	75809	08/02/2011 10:19:54
death.txt	C:\Temp\PCU\death.txt	22497	08/02/2011 10:19:40
diag.txt	C:\Temp\PCU\diag.txt	13867	08/02/2011 10:20:26

รูปที่ 8-15

• แสดงข้อมูลของไฟล์

การแสดงผลข้อมูลของไฟล์นั้นเราจะใช้ฟังก์ชัน `get_file_info()` โดยมีรูปแบบการใช้งาน ดังนี้

```
get_file_info($file, $returned_values)
```

โดยที่ `$file` กำหนดชื่อของไฟล์ที่ต้องการ
`$returned_values` กำหนดค่าที่ต้องการให้โปรแกรมคืนกลับมา
ให้ ค่าปกติ คือ `name`, `server_path`, `size`, `date` โดยให้กำหนดเป็นตัวแปรแบบอาร์เรย์

หมายเหตุ

เราสามารถกำหนดค่า `$returned_values` ได้ ดังนี้

```
$returned_values = array('name', 'server_path', 'size', 'date',  
    'readable', 'writeable', 'executable', 'fileperms')
```

ค่าที่ส่งคืนกลับมาจากฟังก์ชัน `get_file_info()` จะอยู่ในรูปแบบของอาร์เรย์ โดยมีรายละเอียดดังนี้

```
Array (  
    'name'           => ชื่อของไฟล์,  
    'server_path'    => ชื่อพารของไฟล์,  
    'size'           => ขนาดของไฟล์หน่วยเป็นไบต์ (Byte),  
    'date'           => วันที่มีการปรับปรุงแก้ไขล่าสุดอยู่ในรูปแบบของ timestamp,  
    'readable'       => สิทธิ์ในการอ่านไฟล์,  
    'writeable'      => สิทธิ์ในการเขียนไฟล์,
```



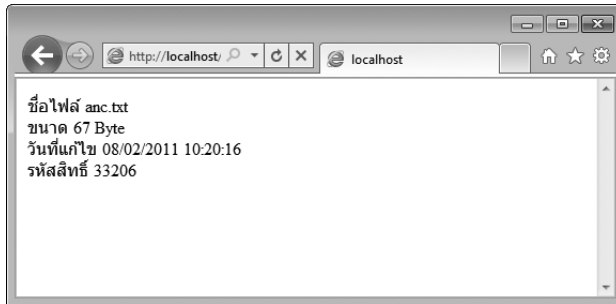
```
'executable'    => สิทธิ์ในการรันไฟล์,
'fileperms'     => รหัสสิทธิ์ของไฟล์ เป็นตัวเลขฐาน 8
)
```

ตัวอย่างการใช้งานฟังก์ชัน `get_file_info()` ให้เราสร้างฟังก์ชันใน Controller ขึ้นมาใหม่ชื่อ `doinfo()` โดยมีรายละเอียดของโค้ด ดังนี้

```
function doinfo()
{
    $path = 'C:\Temp\PCU\anc.txt';
    $returned_values = array('name', 'size', 'date', 'fileperms');
    $files = get_file_info($path, $returned_values);
    if($files)
    {
        echo 'ชื่อไฟล์ ' . $files['name'] . '<br />';
        echo 'ขนาด ' . $files['size'] . ' Byte<br />';
        echo 'วันที่แก้ไข ' . date('d/m/Y H:i:s', $files['date']) . '<br />';
        echo 'รหัสสิทธิ์ ' . $files['fileperms'] . '<br />';
    }else{
        echo 'ไม่สามารถดูข้อมูลได้';
    }
}
```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ <http://localhost/app/index.php/welcome/doinfo> จะปรากฏหน้าจอ ดังนี้





รูปที่ 8-16

• การดูค่า Mime type ของไฟล์

การดูรายละเอียด Mime type ของไฟล์เราจะใช้ฟังก์ชัน `get_mime_by_extension()` โดยฟังก์ชันนี้จะทำการตรวจสอบส่วนขยายของไฟล์ (Extension) หรือสกุลของไฟล์ที่ต่อท้ายเครื่องหมาย '.' เช่น .txt, .rar, .zip ฟังก์ชัน `get_mime_by_extension()` มีรูปแบบการใช้งาน ดังนี้

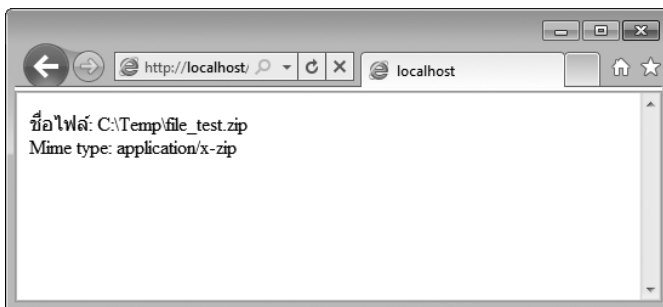
`get_mime_by_extension($file)`

โดยที่ `$file` คือชื่อของไฟล์ที่ต้องการตรวจสอบ mime type

ตัวอย่างการใช้งานฟังก์ชัน `get_mime_by_extension()` ให้เราสร้างฟังก์ชันใน Controller ขึ้นมาใหม่ชื่อ `domime()` โดยเพิ่มโค้ด ดังนี้

```
function domime()
{
    $file = 'C:\\Temp\\file_test.zip';
    echo 'ชื่อไฟล์: ' . $file . '<br />';
    echo 'Mime type: ' . get_mime_by_extension($file);
}
```

จากนั้นที่ Address bar ของเบราว์เซอร์ ให้พิมพ์ `http://localhost/app/index.php/welcome/domime` จะปรากฏหน้าจอ ดังนี้



รูปที่ 8-17

สำหรับในบทนี้เราก็ได้เรียนรู้เกี่ยวกับการใช้งานฟังก์ชันต่าง ๆ ของ CAPTCHA และ File Helper กันครบถ้วนแล้ว เมื่อเราต้องการจะนำไปประยุกต์การใช้งานจริง ไม่ว่าจะเป็นในระบบปฏิบัติการ Windows หรือ Linux สิ่งแรกที่เราควรพิจารณาหรือต้องกำหนดคือ เรื่องของสิทธิหรือ Permission ของไฟล์ต่าง ๆ เพราะ Helper ทั้งสองตัวจะทำงานกับการสร้างไฟล์หรือโฟลเดอร์ หากไม่มีการตรวจสอบค่าของสิทธิหรือ Permission ของไฟล์ โปรแกรมอาจจะไม่ทำงานหรือทำงานผิดพลาดก็เป็นได้



รายชื่อไฟล์ที่แจกผู้อ่านบทที่ 8

1. โฟลเดอร์ captcha_helper ประกอบด้วยโฟลเดอร์ 2 โฟลเดอร์
 - controllers ประกอบด้วยไฟล์ 2 ไฟล์
 - mycaptcha.php
 - welcome.php
 - models ประกอบด้วยไฟล์ 1 ไฟล์
 - captcha_model.php
2. โฟลเดอร์ file_helper ประกอบด้วยโฟลเดอร์ 1 โฟลเดอร์
 - controllers ประกอบด้วยไฟล์ 1 ไฟล์
 - welcome.php