**CodeIgniter: Form APIs, Validations & Pagination Techniques**

## Form APIs

Well, to use form APIs, you must include the “form” in the autoload helpers in your autoload.php configuration file, such as: $autoload['helper'] = array('url','form');

**Note:** You always need to include helpers in the autoload file to access their APIs. For example: If you want to include file uploads in your application, you must include “file” in autoload helper.

Let’s create a simple login form, and add form validation rules. We created the lfy\_users database table with the username and password columns (refer to [previous article](http://www.linuxforu.com/2011/10/lets-play-with-codeigniter/)). I manually added an entry; you can also do this by creating a separate form. Our form is simple to look at; you can add CSS if you want. There will be three files in the MVC structure:

1. Controller File: login.php (/var/www/ci2/application/controllers/login.php)
2. View File: loginview.php (/var/www/ci2/application/views/loginview.php)
3. Model File: loginmodel.php (/var/www/ci2/application/models/loginmodel.php)

### The loginview.php file

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <html>  <head>  <title>Login View </title>  </head>  <body>  <?php echo form\_open('login/check'); ?>  <table border='0'>    <tr align='center'>    <td colspan='2'><b>Welcome To Linux For You</b></td>    </tr>    <tr>    <td>User Name: </td>    <td><input type='text' name='username' id='username' /></td>    </tr>    <tr>      <td>Password: </td>      <td><input type='password' name='password' id='password' /></td>   </tr>   <tr>   <td colspan='2' align='center' style='padding-left:40px'><?php echo form\_submit('submit','Submit') ?>       <input type='button' value=Cancel onClick='window.location.reload();'/></td>   </tr>  </table>  <?php echo form\_close()?>  <?php echo validation\_errors(); ?>  </body>  </html> |
|  |  |

The above code, when simply loaded in the controller file, will give a browser output somewhat like what’s shown in Figure 1.

[](http://cdn.linuxforu.com/wp-content/uploads/2011/12/image1.png)

Figure 1: A simple login page

Here’s the explanation:

1. form\_open('functionname') will create a form with the POST method by default. On submission, it will send the form data to the specified function (here, the check function of the controller file).
2. form\_submit('submit','Submit) will create and perform the submit button action. Arguments passed are type and value.
3. close() marks the end of the form.
4. validation\_errors() will print all validation errors set in the controller file.
5. On cancellation, it will simply reload the page.

### File login.php

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | <?php  class Login extends CI\_Controller {  function index()  {  $this->load->model('loginmodel');  $this->load->view('loginview');  }  function check()  {  $this->load->library('form\_validation');   #1  $this->form\_validation->set\_rules('username','User Name','trim|required'); #2  $this->form\_validation->set\_rules('password','Password','trim|required');  if($this->form\_validation->run()==FALSE)    #3  {  $this->load->view('loginview');  }    $this->load->model('loginmodel');  $userName=$this->input->post('username'); #4  $password=$this->input->post('password');  $userCount=$this->loginmodel->check($userName,$password); #5    if($userCount[0]['count']>0)  {  echo "Login Successful";  }  else if ($userCount[0]['count']==0)  {  echo "Wrong User Name / Password";  }  }// end of funcion check  } //end of class  ?> |

And the explanation for this is as follows:

1. We loaded the form\_validation library manually, to use the built-in validation functions.
2. We set the rules for our data fields. Arguments passed are the field name, error message and validation rules, respectively; the most important is the last one. The required rule makes that field compulsory. You can set multiple rules for a single field using a | operator between them. For example, to specify that a valid email address is required, use required|valid\_email. Explore more built-in validation rules in the documentation of form\_validation.
3. In case form validation fails, the page will simply get reloaded.
4. As explained in the view file source, the form will throw its data to the check function of the controller file. To access that data, use the built-in $this->input->post('fieldname') API of the form helper. Here, they are stored in different variables.
5. Next, control will move to the model file, and it will check for the entered username and password in the database. If the count returns more than one, the username and password is correct, and “Login Successful” will be displayed, else “Wrong user name/password”.

Submitting a form with blank fields yields validation errors, like those in Figure 2.

[](http://cdn.linuxforu.com/wp-content/uploads/2011/12/image2.png)

Figure 2: Validation check

### File loginmodel.php

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  class LoginModel extends CI\_Model {  function check($userName,$password)  {  $query="select count('userName') as count from lfy\_users where userName='$userName' and password='$password'";  $result=$this->db->query($query);  return $result->result\_array();  }  }  ?> |

The check() function will check for the user name and password in the database, as described above.

**Note:** I have used a simple varchar password field in this example, to keep things simple. You should encrypt the password before storing, using MD5, SHA1 types.

## Pagination

Display grids are the simplest way of representing data. But what if there are thousands of records? We use a pagination technique to display a certain amount of records on every page. The built-in CodeIgniter pagination APIs make this quite easy. Here’s an example displaying data of seven users, paginating to two records per page (/var/www/ci2/application/controllers/displaycontroller.php).

### File displaycontroller.php

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | <?php  class DisplayController extends CI\_Controller {  function display($offset=0)  {  $limit=2;             #1  $this->load->model('displaymodel');  $result['contents']=$this->displaymodel->getRows($limit,$offset);  $result['numOfRows']=$this->displaymodel->countRows();      $this->load->library('pagination'); #2      $this->load->library('table');      $config=array(          'base\_url'=>'http://localhost/ci2/index.php/displaycontroller/display',          'total\_rows'=> $result['numOfRows'][0]['count'],          'per\_page'=>$limit,          'num\_links'=>2          );                   #3      $this->pagination->initialize($config);     #4      $result['pagination']=$this->pagination->create\_links();  #5  $this->load->view('displayview',$result);  }  }  ?> |

Here’s the explanation:

1. The idea is to fetch the records according to the limit set (this will also be record\_per\_page) and set the offset value in the URI segment. The initial offset value is 0.
2. We loaded the pagination library.
3. Pagination demands certain parameters to be initialised — we have created an array of such parameters. base\_url is the link to which the page is to be reloaded after getting a hit on any page number; total\_rows refers to the total records; per\_page is the limit of records to show on one page; num\_links is the maximum links allowed for display (after that, “First” and “Last” will appear).
4. We initialised pagination with the above parameters.
5. We fetched the created links into the pagination variable.

You will see something like what is shown in Figure 3 after completing the whole program.



Figure 3: Pagination

Notice the offset value (4 in the URI); it increments by 2 (record\_per\_page) on each page forward, and decreases by 2, going backwards.

### File displayview.php

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <html>  <head>  <title>Display Grid Through Code Igniter</title>  </head>  <body>  Records Found: <?php echo $numOfRows[0]['count']; ?>  <br><br>  <table border=1>  <tr>  <td>S.No </td>  <td>User Name</td>  <td>Password</td>  </tr>  <?php foreach($contents as $users) { ?>  <tr>  <td><?php echo $users['userId']; ?>  <td><?php echo $users['userName'];?></td>  <td><?php echo $users['password'];?></td>  </tr>  <?php } ?>  </table>  <?php if(strlen($pagination)) { ?>  Pages: <?php echo $pagination; }?>  </body>  </html> |

The view file is quite simple. We can add pagination at the end, if it exists. You can apply CSS according to your needs.

### File displaymodel.php

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <?php  class DisplayModel extends CI\_Model{  function getRows($limit,$offset)  {  $query=$this->db->select('userId,userName,password')         ->from('lfy\_users')         ->limit($limit,$offset);  $result=$query->get()->result\_array();  return $result;  }  function countRows()  {  $query="select count(\*) as count from lfy\_users";  $result=$this->db->query($query);  return $result->result\_array();  }  }  ?> |

The first function will fetch and return all data from the lfy\_users table to the controller. The second function counts the total number of records. You can also sort columns similarly; good documentation is available on the CodeIgniter Web portal.

I received many emails in response to the first part of this series, saying that these things can be easily done using simple PHP also. In order that you can understand the logic, I have chosen to use very simple examples. But if you are working on a project of ten thousand lines, and need to change something in your query, I am sure you will waste much time locating the query function, then editing it.

In the MVC-based framework, you just open the model file and make the change. I have worked on a very large CodeIgniter framework-based project, and believe me, it is totally worth your effort. So Google about this more and more, and keep playing with it. Suggestions and queries are always welcome.