In our last series we applied pagination for contents that may be hundreds of records, in this series we will focus on session and Cookie handling.

7. introduction to session: To being able to determine that a sequence of requests are simply coming from the same user, you very often want to maintain state information for a user between requests.
Some applications, such as shopping carts and games, require state in order to function at all, but these are just a subset of the expanse of applications that use state.
Handling state in an application can be a challenge, largely due to the mass of data it is possible to accumulate. If I have a shopping cart application, I need for users to be able to put objects into the cart and track the status of that cart throughout their entire session. PHP offers no data persistence between requests, so you need to tuck this data away someplace where you can access it after the current request is complete.
There are a number of ways to track state. You can use cookies, query string munging, DBM-based session caches, RDBMS-backed caches, application server–based caches, PHP's internal session tools, or something developed in house. With this daunting array of possible choices, you need a strategy for categorizing your techniques. You can bifurcate session-management techniques into two categories, depending on whether you store the bulk of the data client side or server side:

Client-side sessions—Client-side sessions encompass techniques that require all or most of the session-state data to be passed between the client and server on every request. Client-side sessions may seem rather low-tech, and they are some times called heavyweight in reference to the amount of client/server data transmission required. Heavyweight sessions excel where the amount of state data that needs to be maintained is small. They require little to no back-end support. (They have no backing store.) Although they are heavyweight in terms of content transmitted, they are very database/back-end efficient. This also means that they fit with little modification into a distributed system.
Server-side sessions—Server-side sessions are techniques that involve little client/server data transfer. These techniques typically involve assigning an ID to a session and then simply transmitting that ID. On the server side, state is managed in some sort of session cache (typically in a database or file-based handler), and the session ID is used to associate a particular request with its set of state information.
Some server-side session techniques do not extend easily to run in a distributed architecture.

7.1 to apply session handling we need for below steps to take.
7.1.1 New login form and as well changing the database
7.1.2 creation of new controller
7.1.3 creation of required functions in our new controller
7.1.4 creation of new model
7.1.5 creation of new views
7.1.6 changing our existing Home controller

7.2 Now create a new table called user and make relation with posts table
7.2.1 the user table should have these attribute(user_id primary key auto increment, name, lname, user_name, password, email, emage_path);
7.2.2 edit posts table add attribute(user_id) and make the default value 1.

user							
user_id	name	lname	user_name	password	email	emage_path	

posts			
id	title	body	user_id

7.3 We need to create a user for this we need to create new controller, new view and new model.
7.3.1 To create new controller go to C/wamp/www/CI/application/controllers and create new file User.php
In this controller we n need first to load User_model to access it throughout all our functions.
7.3.1 description is there please read it

```
<?php

class User extends CI_Controller{
      function __construct(){
            parent::__construct();
            $this->load->model('User_model');
      }
      /*
       * thisfunction used to loas a view called rigister located in Form
directory of views
       */
      function register(){

            $this->load->view('Form/register');

      }
      //---------------------------------------------------------------------------
-------
      /*
       * this function is to create new user and the form data is sent to this
function from register.php file
       * and send to user table of blog_data database
       */
      function register_user(){
            $data = array(
            'name'=>$this->input->post('name'),
            'lname'=>$this->input->post('lname'),
            'user_name'=>$this->input->post('uname'),
            'password'=>$this->input->post('password'),
            'email'=>$this->input->post('email')

            );
            //echo "<pre>"; print_r($data);echo "</pre>";
            $this->User_model->add_user($data);
            redirect('home/get_posts');

      }
      /*
       * this function load a file called login from the Form Directory.
       */
      function user_login(){
            $this->load->view('Form/login');
      }
      //-----------------------------------------------------------------------------
-------
      /*
       * this function take login data and send to database and than retrive the
desired field and copmare
       * with these data
       */

      public function login() {
            $username = $this -> input -> post('username');
            $password = $this -> input -> post('password');

            if ($this -> input -> post('remember') == 'yes') {
                  // set a cookie with the username. It will expire in 269200
seconds(3 days).
                  $cookie = array('name' => 'username', 'value' => $username,
'expire' => 259200);
                  $this -> input -> set_cookie($cookie);
            } else {
                  // delete the cookie
                  $cookie = array('name' => 'username', 'value' => NULL, 'expire'
=> NULL);
                  $this -> input -> set_cookie($cookie);
            }
            $uid = ($this ->User_model->login($username, $password));
            if (!$uid){
                  redirect(site_url('home/get_posts'));
            }

            // if login successful than send the all data related to the user to
session array
            $this -> session -> set_userdata('uid', $uid);
            $this -> session -> set_userdata('user', $this->
User_model->get_userdata($uid));
            $this -> session -> set_userdata('login', true);
            //print_r($this->session->all_userdata());
            redirect($this -> session -> userdata('last_visited'));
      }
      //-----------------------------------------------------------------------------
-------
      // this function is called by log out button and delete the session array
      public function logout() {
            $this -> session -> unset_userdata('login');
            $this -> session -> unset_userdata('user');
            $this -> session -> unset_userdata('uid');

            redirect($this -> session -> userdata('last_visited'));
      }


}
```

7.3.2 now we need to create a model called User_model, to create this model go to C/wamp/www/CI/application/models directory and create new files called User_model.php and create function for this model as below.

```
<?php
class User_model extends CI_Model{

      /*
       * this function is for creating new User
       */
      function add_user($data){

            $this->db->insert('user',$data);
      }
      /*
       * this function is to compare user name from form comming with existing
values in th database
       */

      function login($username, $password){

            $array = array('user_name' => $username, 'password' => $password) ;
            $this->db->where($array);
            $this -> db -> select('user_id');
            $query = $this -> db -> get('user');
            if ($query -> num_rows() > 0) {
                  return $query -> first_row()->user_id;
            } else{
                  return 0;
            }

      }
      /*
       * this function is to return all user data
       */
      function get_userdata($uid) {
            $this -> db -> where('user_id', $uid);
            $query = $this -> db -> get('user');
            if ($query -> num_rows() > 0) {
                  return $query -> first_row();
            } else
                  return false;
      }
}
```

7.3.3 now we need to create a view for adding new user, for this we need to structure our view on a way to have clear structure, therefore we creating new folder Called Form in our views directory. In this directory we now creating a file called register.php and putting all inputs that is required for a user registration.

```
<h1> Add User </h1>
<?php echo form_open('User/register_user');?>
<table>
      <tr>
            <td>Name:</td>
            <td><input type="text" name="name"/></td>

      </tr>
      <tr>
            <td>Last Name:</td>
            <td><input type="text" name="lname"/></td>

      </tr>
      <tr>
            <td>User Name: </td>
            <td><input type="text" name="uname"/></td>

      </tr>
      <tr>
            <td>Password: </td>
            <td><input type="password" name="password"/></td>

      </tr>
      <tr>
            <td>Email </td>
            <td><input type="email" name="email"/></td>

      </tr>
      <tr>
            <td>Image </td>
            <td><input type="file" name="myImage" /></td>

      </tr>


</table>

<input type="submit" value="Save Changes" />
```

7.3.4 now we need to create a form for login, so go to C/wamp/www/CI/application/views/Form and create login.php file put reguired inputs on it as below.

```
<h1> Add Post Data </h1>
<?php echo form_open('User/login');?>
<table>
      <tr>
            <td>User Name: </td>
            <td><input type="text" name="username"/></td>

      </tr>
      <tr>
            <td>Password: </td>
            <td><input type="password" name="password"/></td>

      </tr>
      <tr>
            <td>Remember: </td>
            <td><input type="checkbox" name="remeber"/></td>

      </tr>

</table>

<input type="submit" value="Log IN" />
```

7.3.5 now we need to change our existing Home controller. The aim is when we adding a post, we need the id of a user that is registered and login should be putted in to the user table so we editing the add function of our Home controller as below. To do this go to C/wamp/www/CI/application/controllers/Home.php

```
class Home extends CI_Controller {

      function __construct() {
            parent::__construct();
            $this->load->model('Posts_model');
      }

      //------------------------------------------------------------------
      function index(){

            echo "Hello CodeIgniter";
      }
      //------------------------------------------------------------------
      function dynamic_view(){
            $data['title'] = 'Dynamic View';
            $data['body'] = 'A view for Dynamic Contents!';
            $this->load->view('some_view',$data);
      }
      //------------------------------------------------------------------
      function get_posts(){
// we need to set last visited as current URL to session
            $this -> session -> set_userdata('last_visited', current_url());
            $this->load->library('pagination');
            $offset=$this->uri->segment(3);
            $config['base_url'] = base_url().'index.php/home/get_posts';
            $config['total_rows'] =$this->Posts_model->count_posts();
            $config['per_page'] = 3;
            $config['uri_segment'] = 3;
            $this->pagination->initialize($config);
            $data['posts']=$this->Posts_model->get_posts($config['per_page'],
$offset);
            //$data['posts']=$this->Posts_model->get_posts();
            $this->load->view('something',$data);
      }
      //------------------------------------------------------------------
      function details($id){
            $data['post']=$this->Posts_model->get_post($id);
            $this->load->view('show_d',$data);
      }
      //------------------------------------------------------------------
      function add_post(){
            $this->load->view('new_post');
      }
      //------------------------------------------------------------------
      function add(){
            $user = $this->session->userdata('user');
            $user_id=2;
// if user is loged in than the user id is equal to the user id of the loged in
// use else the user 1 will be inserted to database
            if ($this->session->userdata('login')) {
            $user_id=$user->user_id;
            }else{
                  $user_id=1;
            }
            $data = array(
                  'title' =>$this->input->post('title') ,
                  'body' =>$this->input->post('body'),
                  'user_id'=> $user_id
                  );
            $this->Posts_model->add_post($data);
            redirect('home/get_posts');

      }
      //-----------------------------------------
      function delete($id){
            $this->Posts_model->delete_post($id);
          redirect('home/get_posts');
      }
      //----------------------------------------------
      function update_post($id){
            $data['post']=$this->Posts_model->get_post($id);
            $this->load->view('update_post',$data);
      }
      //------------------------------------------------------------------
      function update($id){
            $data = array(
                  'title' =>$this->input->post('title') ,
                  'body' =>$this->input->post('body')
            );
            $this->Posts_model->update_post($id,$data);
            redirect('home/get_posts');

      }
}
```