Министерство науки и высшего образования Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: «Веб-технологии»

Лабораторная работа № 2

«Исследование возможностей хранения данных на стороне сервера. Работа с файлами. Работа с реляционными СУБД»

Выполнила

ст. гр. ИС/б-17-2-о

Кайда Е.Н.

Проверил:

Овчинников А.Л.

Севастополь

2020

Цель работы

Изучить возможности хранения данных на стороне сервера: работу c файлами и СУБД MySQL из PHP, приобрести практические навыки организации хранения данных на стороне сервера в файлах, в базах данных MySQL, а также овладение навыками постраничного вывода данных.

Постановка задачи

1. Разработать базовый класс BaseActiveRecord для работы с базой данных, который реализует паттерн ActiveRecord (данный класс разместить в папке /my\_site/app/core).

2. Для всех моделей, которые будут использоваться при выполнении данной лабораторной работы, создать классы, наследующие BaseActiveRecord. Для каждого из классов определить все поля и названия таблиц (данные классы необходимо разместить в папке /my\_site/app/models).

3. Создать новую страницу "Гостевая книга". Страница должна содержать форму ввода (Фамилия, Имя, Отчество, E –mail, Текст отзыва), а также таблицу сообщений, оставленных пользователями. Сообщения в таблице должны располагаться в порядке убывания даты добавления сообщения. Для хранения сообщений пользователей использовать текстовый файл messages.inc, содержащий разделенные символом «;» данные: Дату сообщения, ФИО, E–mail и Текст отзыва (для получения текущей даты сервера возможно использовать PHP функцию *date('d.m.y')).*

4. Реализовать страницу "Загрузка сообщений гостевой книги", содержащую форму загрузки подготовленного заранее файла messages.inc на сервер.

5. Реализовать на странице "Тест по дисциплине" сохранение ответов пользователей и правильности ответов в разработанную таблицу базы данных MySQL, с возможностью просмотра сохраненных данных (дата, ФИО, ответы, верно/неверно).

6. Разработать страницу «Редактор Блога», позволяющую добавлять записи Блога. Страница должна содержать форму добавления записи Блога и список выдаваемых постранично записей отсортированных в порядке убывания даты. Форма добавления должна содержать поля ввода:

* Тема сообщения – поле ввода однострочного текста (заполнение обязательно);
* Изображение – поле ввода файла (заполнение не обязательно);
* Текст сообщения – поле ввода многострочного текста.

Данные хранить в разработанной таблице базы данных MySQL. Валидацию данных осуществлять с использованием класса FormValidation.

7. Разработать страницу «Мой Блог», содержащую упорядоченные в порядке убывания даты добавления, выдаваемые постранично данные:

* Дата и время сообщения;
* Тема сообщения;
* Изображение;
* Текст сообщения.

Данные извлекать из таблицы базы данных MySQL.

8. Реализовать возможность добавления записей на страницу «Мой Блог» из файла формата CSV, содержащего следующие поля: title, message, author, created\_at. Например: "тема 1","сообщение 1","Vasiliy","2019-01-01 14:00". Для этого необходимо разработать страницу «Загрузка сообщений блога», содержащую форму загрузки файла формата CSV. Добавление записей из файла в БД осуществлять с использованием подготавливаемых запросов.

ход работы

1. Разработаем базовый класс BaseActiveRecord:

abstract class BaseActiveRecord{

    public static $pdo;

    protected static $tablename;

    protected static $dbfields = array();

    public function \_\_construct() {

        if (!static::$tablename){

            echo " if (!static::\$tablename)";

            return ;

        }

        static::setupConnection();

        static::getFields();

    }

Автораспознание полей таблицы по ее названию

    public static function getFields() {

        $stmt = static::$pdo->query("SHOW FIELDS FROM ".static::$tablename);

        for ($i=0;$row = $stmt->fetch(); $i++ ){

            static::$dbfields[$i] = $row['Field'];

        }

    }

    public static function getTableName(){

        return static::$tablename;

    }

    public static function setupConnection() {

        if (!isset(static::$pdo)) {

        $eror = false;

        try {

            $config = require 'app/config/db.php';

            static::$pdo = new PDO('mysql:host='.$config['host'].';dbname='.$config['name'].'', $config['user'], $config['password']);

        } catch (PDOException $ex) {

            die("Ошибка подключения к БД: $ex");

            echo "Ошибка подключения к БД: $ex";

         }

        }

    }

Функция поиска по id

    public static function find($id){

        $sql = "SELECT \* FROM ".static::$tablename." WHERE id=$id";

        $stmt = static::$pdo->query($sql);

        $row = $stmt->fetchAll(PDO::FETCH\_ASSOC);

        if (!$row) {

            return null;

        }

        return $row ;

    }

Функция для выбора всех строк из таблицы

    public static function findAll($fields = null) {

        if ($fields == null){

            $sql = "SELECT \* FROM ".static::$tablename;

            $stmt = static::$pdo->query($sql);

            $stmt = $stmt->fetchAll(PDO::FETCH\_ASSOC);

            return $stmt;

        }

    }

Функция вставки строк

    public function save($values) {

        //Запрос должен выглядить так:

        //INSERT INTO nameTable (field1, field2, field3) VALUES (?,?,?);

        $fields = "";

        $val = "";

        foreach(static::$dbfields as $temp){

            $fields .= "`".$temp."`,";

            $val = $val.'?,';

        }

        $fields = rtrim($fields, ",");

        $val = rtrim($val, ",");

        $query = "INSERT INTO ".static::$tablename."($fields) VALUES ($val)";

//подставление значений

        $stmt = static::$pdo->prepare($query);

        try {

            static::$pdo->beginTransaction();

            foreach ($values as $row){

                $stmt->execute($row);

            }

            static::$pdo->commit();

        }catch (PDOException $e){

            static::$pdo->rollback();

            throw $e;

        }

    }

Функция изменения записи

    public function update( $params){

       //"UPDATE users SET name=:name, surname=:surname, sex=:sex WHERE id=:id";

       //"UPDATE users SET name='lena', surname='kaida', sex='w' WHERE id=:id";

        $strVars = '';

        foreach(static::$dbfields as $field){

            $strVars .= $field."=:".$field.",";

        }

        $strVars = rtrim($strVars, ",");

        $query= "UPDATE ".static::$tablename." SET ".$strVars." WHERE `id`=:id";

        $stmt = static::$pdo->prepare($query);

        $stmt->execute($params);

    }

Функция удаления записи

    public function delete(){

        $sql = "DELETE FROM ".static::$tablename." WHERE ID=".$this->id;

        $stmt = static::$pdo->query($sql);

        if($stmt){

        return $stmt->fetch(PDO::FETCH\_ASSOC);

        }else{

        print\_r(static::$pdo->errorInfo());

        }

    }

}

1. Создадим новую страницу "Гостевая книга" с формой для отправки отзыва и таблицей сообщений, которые считываются из файла messages.inc:

Было создано представление:

<?php

//Обработка отправки формы, где мы вызываем метод сохранения данных контроллера

$values = [];

     if ($\_SERVER["REQUEST\_METHOD"] == "POST"){

         $valid->Validate($\_POST);

         if (!$valid->checkErrors($errors)) $values = $\_POST;

         else{

              $controller->save($\_POST);

              if($valid->checkErrors($errors))

               echo "<div id='resultWindiw' style=\"color: green; font-size: 30px;\">Форма успешно отправлена</div>";

         }

     }

?>

<section>

     <h1>Здесь вы можете оставить свой отзыв</h1>

     <form  action="" method=POST >

          <p><input type="text" id="name" name="name" placeholder="ФИО" data-toggle="popover" onblur="validate(this)"

              value="<?php if (array\_key\_exists('name', $values)) echo $values['name'] ?>"

              data-content="<?php echo $rules['name'] ?>">

              <br><pre><?php echo $valid->getError('name') ?><br></pre><br>

         </p>

          <p><input type="text" id="mail" name="email" onblur="validate(this)" placeholder="Ваш Еmail"

             value="<?php if (array\_key\_exists('email', $values)) echo $values['email'] ?>"

             data-toggle="popover" data-content="<?php echo $rules['email'] ?>">

             <pre><?php echo $valid->getError('email')?><br></pre><br>

        </p>

          <p><textarea id="message" name="message" onblur="validate(this)" data-toggle="popover" placeholder="Ваше сообщение"

             value="<?php if (array\_key\_exists('message', $values)) echo $values['message'] ?>"

             data-content="<?php echo $rules['message'] ?>"></textarea>

             <br><pre><?php echo $valid->getError('message') ?><br></pre> <br>

        </p>

          <input type="submit"  id="submit" value="Отправить" >

          <input type="reset" id="reset" value="Очистить" >

     </form>

     <br/><h1>Отзывы</h1>

     <table class="comments">

          <tbody>

//Вывод прочитанных комментариев из переменной $comments в таблицу

               <?php

                    if($comments == null): ?>

                         <h2>Нет отзывов</h2>

               <?php else:

                         foreach ($comments as $value) {

                              echo "<tr>";

                              $date = DateTime::createFromFormat('d.m.Y H:i:s', $value['date']);

                              echo "<td class=\"info\"><p>". $date->format('d.m.Y H:i:s')."</p>";

                              echo "<h4>" .$value['fio']."</h4>";

                              echo "<p>" .$value['email']."</p></td>";

                              echo "<td class=\"text\"><p>".$value['msg']."</p>";

                              echo "</tr>";

                         }

                    endif;

               ?>

          </tbody>

     </table>

</section>

Контроллер для этой страницы:

class GuestController extends Controller{

    public function  bookAction(){

        $this->data['controller'] = $this;

        $this->view->render('Гостевая книга', $this->data);

    }

//Метод сохранения отправленного отзыва, где вызывается метод модели для сохранения данных в файл

    public function save($post\_array){

        $respons = [

            $post\_array['name'],

            $post\_array['email'],

            date('d.m.Y H:i:s'),

            $post\_array['message']

        ];

        $this->model->sendRespons("messages.inc", $respons);

    }

}

Модель:

class Guest extends Model{

    public function \_\_construct($action){

        parent::\_\_construct($action);

    }

//Создаем массив данных для view (считываем отзывы из файла)

    public function book(){

        $this->validator = new ValidGuestBook();

        return [

           'valid' => $this->validator,

           'rules' => $this->validator->getRules(),

           'errors' => $this->validator->getErrors(),

           'values'=> $this->validator->getErrors(),

           'comments' => $this->readComments("messages.inc"),

       ];

    }

//Метод считывания отзывов из файла

    public function readComments($nameFile){

         $file = fopen("public/files/$nameFile", "r");

         $comments = [];

         for( $i=0; !feof($file) ; $i++){

              $str = fgets($file);

              if(empty($str)) continue;

              $spitedStr = explode(';', $str);

              $temp = [

                   'fio' => trim($spitedStr[0]),

                   'email' => trim($spitedStr[1]),

                   'date' =>  trim($spitedStr[2]),

                   'msg' => trim($spitedStr[3])

              ];

              $comments[$i] = $temp;

         }

         fclose($file);

//сортируем отзывы по дате добавления (сначала новые)

usort($comments, function($first, $second){

               $first = DateTime::createFromFormat('d.m.Y H:i:s', $first['date']);

               $second = DateTime::createFromFormat('d.m.Y H:i:s', $second['date']);

               return $first < $second;

         });

         return $comments;

    }

//Метод сохранения отзыва в файл

    public function sendRespons($nameFile, $respons){

          $file = fopen("public/$nameFile", "a");

          foreach( $respons as $temp){

               fwrite($file, $temp."; ");

          }

          fwrite($file, "\n");

          fclose($file);

    }

}

1. Реализована страница "Загрузка сообщений гостевой книги", содержащая форму загрузки файла messages.inc на сервер.

Представление страницы "Загрузка сообщений гостевой книги":

<section>

    <h1>Загрузка файла с отзывами</h1>

    <?php

        //обработка запроса на отправку файла (вызов функции модели на отправку файла)

        if ($\_SERVER["REQUEST\_METHOD"] == "POST"){

            echo "<div style=\"color: blue;\">".$model->loadGuestBook($\_FILES, "userFile")."</div>";

            $new\_url = 'http://mysite.com/guestBook';

            header('Location: '.$new\_url);

        }

    ?>

    <br/>

    <form enctype="multipart/form-data" action="" method="POST">

        <input type="hidden" name="30000" value="30000" />

        <input name="userFile" type="file">

        <br/>

        <input type="submit" value="Загрузить">

    </form>

</section>

Контроллер:

class CommentsController extends Controller{

    public function  uploadAction(){

        $vars=[

            'model' => $this->model

        ];

        $this->view->render('Загрузка отзывов', $vars);

    }

}

Модель:

class Comments extends Model{

    public function \_\_construct($action){

        parent::\_\_construct($action);

    }

    public function upload(){

    }

//метод для отправки на сервер файла с отзывами

    public function loadGuestBook($FILES, $nameField){

        if (!empty($FILES)){

            $file = "D:/web/websitePHP/public/".$\_FILES[$nameField]['name'];

            if($\_FILES[$nameField]['error'] == UPLOAD\_ERR\_OK){

                move\_uploaded\_file($\_FILES[$nameField]['tmp\_name'], $file);

                return "Загрузка прошла успешно";

            }

        }

        return "Ошибка загрузки";

    }

}

1. Реализуем на странице "Тест по дисциплине" сохранение ответов пользователей и правильности ответов в таблицу базы данных MySQL, с возможностью просмотра сохраненных данных.

В представлении была добавлена обработка результатов:

if ($\_SERVER["REQUEST\_METHOD"] == "POST"):

        $controller->sendResults($\_POST);

Для этого в контроллере были созданы дополнительные методы:

class ExamController extends Controller{

    public $examDB;

    public function  testAction(){

//создание модели БД для этой сущности

        $examDB = new ExamRecord();

        $this->data['controller'] = $this;

        $this->view->render('Тест', $this->data);

    }

//метод обработки и отправки в БД результатов теста

    public function sendResults($post\_array){

        $this->data['valid']->Validate($post\_array);

        $errors = $this->model->validator->getErrors();

        if($errors['name'] != null) return;

        function getAnswer3($post\_array){

            if (array\_key\_exists('question3', $post\_array)){

                $answer3 = "";

                foreach($post\_array['question3'] as $val){

                    $answer3 = $answer3."  ".$val;

                }

                return $answer3;

            }

            return null;

        }

        $ansswers = [

            'id' => null,

            'fio' => (array\_key\_exists('name', $post\_array) ? trim($post\_array['name']) : null),

            'date' => date('Y-m-d H:i:s'),

            'course' => (array\_key\_exists('course', $post\_array) ? $post\_array['course'] : null),

            'answer1' => (array\_key\_exists('question1', $post\_array) ? $post\_array['question1'] : null),

            'flag1' => (($errors['question1'] === null) ? 1 : 0),

            'answer2' => (array\_key\_exists('question2', $post\_array) ? $post\_array['question2'] : null),

            'flag2' => (($errors['question2'] === null) ? 1 : 0),

            'answer3' => getAnswer3($post\_array),

            'flag3' => (($errors['question3'] === null) ? 1 : 0),

        ];

//Вызов метода сохранения данных в БД

        $examDB->insert($ansswers);

    }

}

Модель БД для сущности:

class ExamRecord extends BaseActiveRecord{

    protected static  $tablename = "test";

    protected static $dbfields = [];

    public static  $pdo;

    public function \_\_construct(){

        parent::\_\_construct();

    }

    public function insert($val\_arr){

        $val = [

            array\_values($val\_arr)

        ];

//вызов метода родителя для вставки данных в таблицу БД

        parent::save( $val);

    }

После ввода всех ответов на странице теста и отправки данных появляется окно с ответами и результатами (рис. 1). На рисунке 2 показана созданная запись в БД.

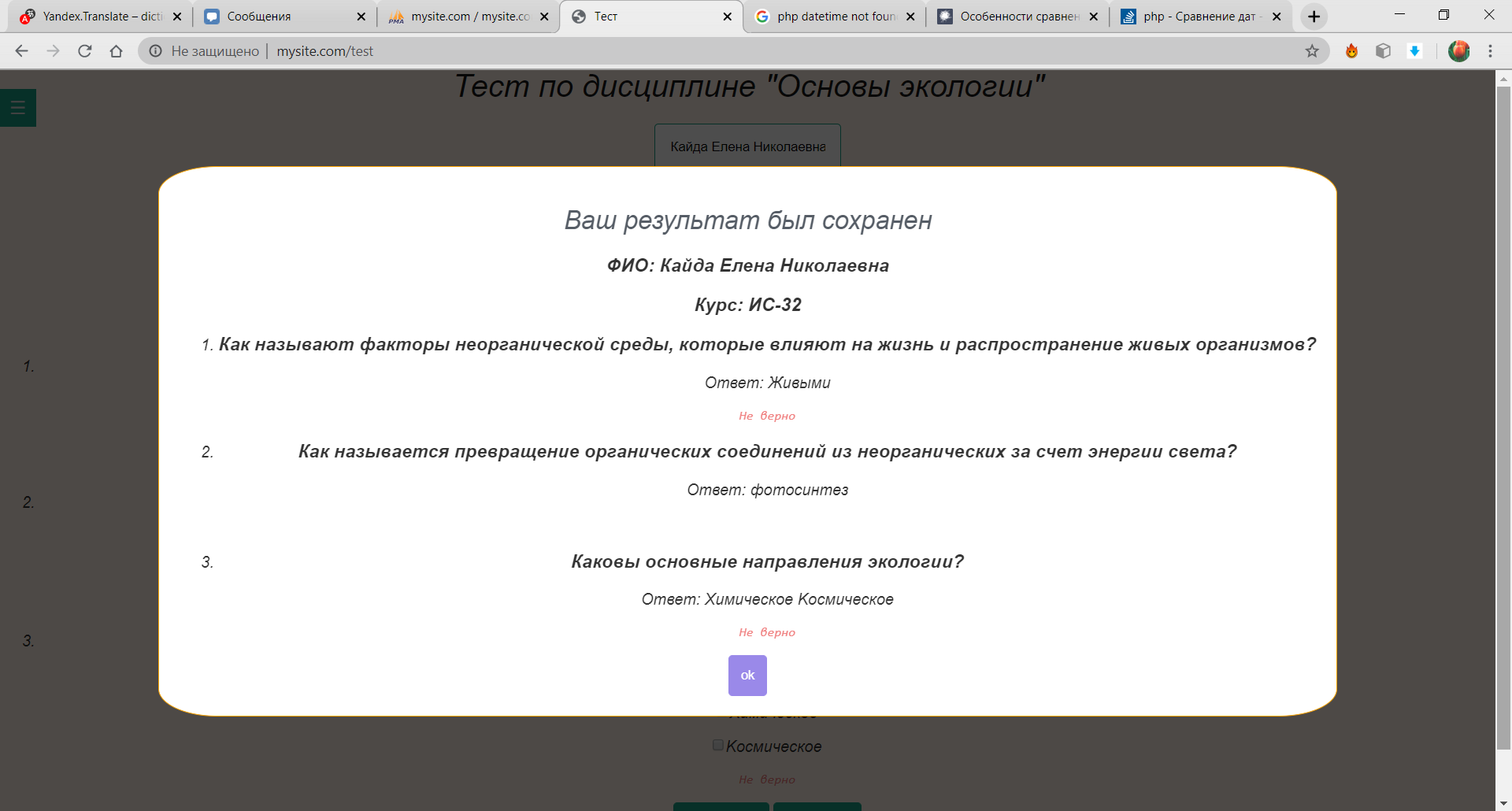


Рисунок 1 – Результат теста

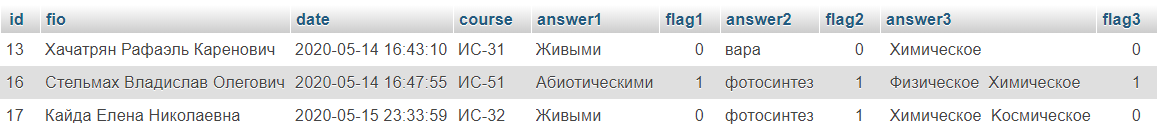


Рисунок 2 – Созданная запись в БД

1. Разработана страница «Редактор Блога», позволяющая добавлять записи Блога через форму. Данные сохраняются в таблице базы данных MySQL.

Представление страницы:

<section>

    <?php

        $values = [];

        if ($\_SERVER["REQUEST\_METHOD"] == "POST"){

//метод сохранения новой записи блога

            $controller->save($\_POST, $\_FILES);

            if (!$valid->checkErrors($errors)) $values = $\_POST;

        }

    ?>

    <h1>Добавление записи Блога</h1>

    <form enctype="multipart/form-data"  action="" method=POST >

        <p><input type="text" id="topic" name="topic" placeholder="Тема сообщения" data-toggle="popover"

            >

            <br><pre><?php echo $valid->getError('topic') ?><br></pre><br>

        </p>

        <p><textarea id="message" name="message" data-toggle="popover" placeholder="Ваше сообщение..."

            ></textarea>

            <br><pre><?php echo $valid->getError('message') ?><br></pre> <br>

        </p>

        <p>

            <input type="hidden" name="30000" value="30000" />

            <input name="userFile" type="file" accept="image/\*">

            <br><pre><?php echo $valid->getError('userFile') ?><br></pre> <br>

        </p>

        <input type="submit"  id="submit" value="Отправить" >

        <input type="reset" id="reset" value="Очистить" >

    </form>

</section>

Контроллер:

class BlogController extends Controller{

    public function  sendAction(){

        $this->data['controller'] = $this;

        $this->view->render('Редактор Блога', $this->data);

    }

//метод сохранения новой записи блога

    public function save($post\_array, $files\_array){

        $blogDB = new BlogRecord();

        $post\_array['userFile'] = $files\_array;

        $this->data['valid']->Validate($post\_array);

        $errors = $this->model->validator->getErrors();

        if(!$this->data['valid']->checkErrors()) return;

        if($files\_array['userFile']['error'] == UPLOAD\_ERR\_OK){

            $upload\_image = $files\_array["userFile"]["name"];

            $folder="D:/web/websitePHP/public/assets/img/";

            move\_uploaded\_file($files\_array["userFile"]["tmp\_name"], $folder.$upload\_image);

        }

        $data = [

            'id' => null,

            'topic' => (array\_key\_exists('topic', $post\_array) ? trim($post\_array['topic']) : null),

            'message' => (array\_key\_exists('message', $post\_array) ? trim($post\_array['message']) : null),

            'img name' => ($files\_array['userFile']['error'] == UPLOAD\_ERR\_OK) ?

                            $files\_array['userFile']['name'] : null,

            'img src' => ($files\_array['userFile']['error'] == UPLOAD\_ERR\_OK) ?

                            '/public/assets/img/': null,

            'date' => date('Y-m-d H:i:s')

        ];

//вызов метода сохранения от БД сущности

        $blogDB->insert($data);

    }

}

БД сущности:

class BlogRecord extends BaseActiveRecord{

    protected static  $tablename = "blog";

    protected static $dbfields = [];

    public static  $pdo;

    public function \_\_construct(){

        parent::\_\_construct();

    }

    public function insert($val\_arr){

        $val = [

            array\_values($val\_arr)

        ];

        parent::save( $val);

    }

1. Разработаем страницу «Мой Блог», содержащую упорядоченные в порядке убывания даты добавления, выдаваемые постранично сообщения. Данные извлекать из таблицы базы данных MySQL.

Представление страницы:

<section>

     <h1>Мой Блог</h1>

     <table class="comments">

          <tbody>

               <?php

                    if(empty($rows)): ?>

                         <h2>Нет записей</h2>

               <?php else:

                         foreach ($rows as $temp) {

                              echo "<tr>";

                              $date = DateTime::createFromFormat('Y-m-d H:i:s', $temp['date']);

                              echo "<td class=\"info\"><p>". $date->format('d.m.Y  H:i:s')."</p>";

                              echo ($temp['img src'] == null) ? "</td>" : "<img src='" .$temp['img src'].$temp['img name']."'></td>";

                              // echo "<script> console.log(\"<img src='" .$temp['img src'].$temp['img name']."'></td>\") </script>";

                              echo "<td class=\"text\"><h3>".$temp['topic']."</h3>";

                              echo "<p>".$temp['message']."</p></td>";

                              echo "</tr>";

                         }

                    endif;

               ?>

          </tbody>

     </table>

//вывод ссылок страниц

     <?php

     foreach($pages as $i){

          echo $i;

     }

     ?>

     <br/>

</section>

Контроллер:

class DiaryController extends Controller{

    public function  blogAction(){

        $this->data['controller'] = $this;

        $this->view->render('Мой Блог', $this->data);

    }

}

Модель:

class Diary extends Model{

    public function \_\_construct($action){

        parent::\_\_construct($action);

    }

    public function blog(){

        $diaryDB = new DiaryRecord();

//вызов метода формирования ссылок на страницы

        return $diaryDB->linksPages();

    }

}

Модель БД сущности

class DiaryRecord extends BaseActiveRecord{

    protected static  $tablename = "blog";

    protected static $dbfields = [];

    public static  $pdo;

    public function \_\_construct(){

        parent::\_\_construct();

    }

//метод формирования сраниц

    public function linksPages(){

        // количество записей, выводимых на странице

        $per\_page=5;

        // получаем номер страницы

        $page= (int)(isset($\_GET['page']) ? ($\_GET['page']-1) : 0);

        // вычисляем первый операнд для LIMIT

        $start=abs($page\*$per\_page);

        // выполняем запрос и выводим записи

        $query = "SELECT \* FROM ".static::$tablename." ORDER BY `".static::$tablename."`.`date` DESC LIMIT $start, $per\_page";

        $rows = static::$pdo->query($query);

        $pages = [];

        // выводим ссылки на страницы:

        $query = "SELECT count(\*) FROM ".static::$tablename;

        $total\_rows = static::$pdo->query($query)->fetchColumn();

        // Определяем общее количество страниц

        $num\_pages = ceil($total\_rows/$per\_page);

        for($i=1;$i <= $num\_pages; $i++) {

            // текущую страницу выводим без ссылки

            if ($i-1 == $page) {

                $pages[$i] = "$i ";

            } else {

                $pages[$i] = "<a href='blog?page=".$i."'>".$i."</a> ";

            }

        }

        return [

            'rows' => $rows,

            'pages' => $pages

        ];

    }

}

1. Разработана страница «Загрузка сообщений блога», содержащая форму загрузки файла формата CSV, которая добавляет запись на страницу «Мой Блог» из файла формата CSV. Добавление записей из файла в БД осуществлять с использованием подготавливаемых запросов.

Представление:

<section>

    <h1>Загрузка файла с сообщениями Блога</h1>

    <?php

        if ($\_SERVER["REQUEST\_METHOD"] == "POST"){

            echo "<div style=\"color: blue;\">".$controller->saveRecords("userFile")."</div>";

        }

    ?>

    <br/>

    <h3> Файл CSV </h3>

    <form enctype="multipart/form-data" action="" method="POST">

        <input type="hidden" name="300000" value="30000" />

        <input name="userFile" type="file">

        <br/>

        <input type="submit" value="Загрузить">

    </form>

</section>

Контроллер:

class RecordsController extends Controller{

    public function  uploadAction(){

        $this->data['controller'] = $this;

        $this->view->render('Загрузка Блога', $this->data);

    }

//функция сохранения записей

    public function saveRecords($nameField){

        if (!empty($\_FILES)){

            $file = "D:/web/websitePHP/public/files".$\_FILES[$nameField]['name'];

            if($\_FILES[$nameField]['error'] == UPLOAD\_ERR\_OK){

                move\_uploaded\_file($\_FILES[$nameField]['tmp\_name'], $file);

//вызов функции модели сохранения записей из файла в БД

                return $this->model->save($file);

            }

        }

        return "Ошибка загрузки";

    }

}

Модель:

class Records extends Model{

    public $validator;

    public function \_\_construct($action){

        parent::\_\_construct($action);

    }

    public function upload(){

        $this->validator = new ValidBlog();

    }

    public function save($pathFile){

        $recordsBD = new RecordsRecord();

        $result = [];

        $file = fopen($pathFile, "r");

        $i = 0;

            while (($data = fgetcsv($file, 1000, ";")) !== FALSE) {

                $row = [

                    'id' => null,

                    'topic' => $data[0],

                    'message' => $data[1],

                    'img name' => $data[2],

                    'img src' => $data[3],

                    'date' => $data[4]

                ];

                $this->validator->Validate($row);

                if ($this->validator->checkErrors()) $result[$i++] = array\_values($row);

            }

            fclose($file);

            if (empty($result)) return "Данные не были сохранены";

//вызов функции записи в БД

            $recordsBD->save($result);

            return "Данные сохранены";

    }

}

Модель БД для сущности:

class RecordsRecord extends BaseActiveRecord{

    protected static  $tablename = "blog";

    protected static $dbfields = [];

    public static  $pdo;

    public function \_\_construct(){

        parent::\_\_construct();

    }

}

Подробный код всех представлен на сайте GitHub (Приложение А).

Вывод

В ходе выполнения лабораторной работы были изучены возможности хранения данных на стороне сервера: работа c файлами и СУБД MySQL из PHP, приобретены практические навыки организации хранения данных на стороне сервера в файлах, в базах данных MySQL, а также овладели навыками постраничного вывода данных.

Приложение А

1. Код сайта - https://github.com/kaidalena/web\_lr\_semestr\_6