Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №2

Исследование возможностей хранения данных на стороне сервера.

Работа с файлами. Работа с реляционными СУБД

по дисциплине «Веб-технологии»

Выполнил:

Студент группы ИС/б 17-2-о

Черняев Н.Г.

Проверил:

Забаштанский А.К.

г. Севастополь 2020

1. 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.

2. Постановка задачи

1. Разработать базовый класс BaseActiveRecord для работы с базой данных, который реализует паттерн ActiveRecord.

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.

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.

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, разработанного при выполнении ЛР №5.

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. Формат постраничного вывода определяется в соответствии с вариантом задания.

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. Валидацию данных осуществлять с использованием класса FormValidation, разработанного при выполнении ЛР №5.

3. ход работы

1. Разработаем базовый класс BaseActiveRecord для работы с базой данных, который реализует паттерн ActiveRecord. Код данного класса представлен ниже:

<?php

class BaseActiveRecords {

public static *$pdo*;

protected static *$tablename*;

protected static *$dbfields* = array();

public function \_\_construct() {

if (!static::*$tablename*) {

return;

}

static::*setupConnection*();

static::*getFields*();

}

public static function getFields() {

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

while ($row = $stmt->fetch()) {

static::*$dbfields*[$row['Field']] = $row['Type'];

}

}

public static function setupConnection() {

if (!isset(static::*$pdo*)) {

try {

static::*$pdo* = new PDO("pgsql:dbname=web2; host=localhost;", "postgres", "");

} catch (PDOException $ex) {

die("Ошибка подключения к БД: $ex");

}

}

}

public static function find($id) {

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

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

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

if (!$row) {

return null;

}

$ar\_obj = new static();

foreach ($row as $key => $value) {

$ar\_obj ->$key = $value;

}

return $ar\_obj;

}

public static function findAll() {

static::*setupConnection*();

static::*getFields*();

$result = [];

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

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

while($row = $stmt->fetch(PDO::*FETCH\_ASSOC*)) {

array\_push($result, $row);

}

return $result;

}

public static function findByPage($offset, $rowsPerPage) {

static::*setupConnection*();

$result = [];

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

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

while($row = $stmt->fetch(PDO::*FETCH\_ASSOC*)) {

array\_push($result, $row);

}

return $result;

}

public static function getCount() {

static::*setupConnection*();

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

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

$result = $stmt->fetch(PDO::*FETCH\_ASSOC*);

return current($result);

}

public function save($data) {

static::*setupConnection*();

$values = implode("', '", $data);

$values = '\'' . $values . '\'';

$fields = implode(",", static::*$dbfields*);

$tablename1 = static::*$tablename*;

$sql = "INSERT INTO $tablename1 ($fields) VALUES ($values)";

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

if ($stmt) {

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

} else {

throw new Exception();

}

}

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());

}

}

}

2. Создадим новую страницу "Гостевая книга". Страница должна содержать форму ввода, 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.

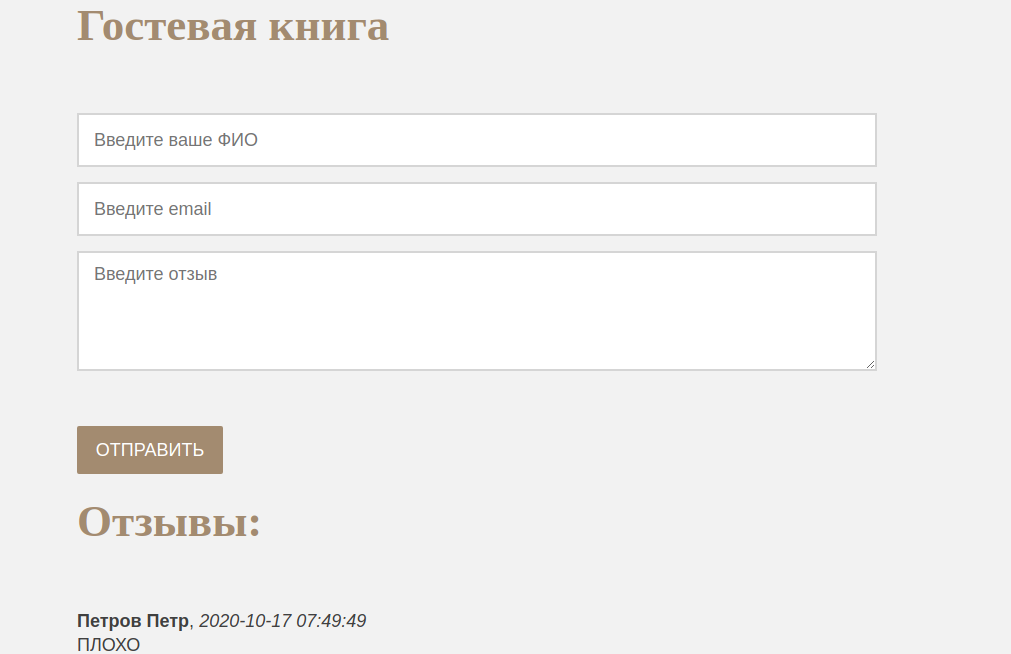


Рисунок 1 – Страница “Гостевая книга”

Контроллер GuestBookController.php:

<?php

class GuestBookController extends Controller {

function \_\_construct() {

$this->model = new GuestBookModel();

$this->view = new View();

}

function indexAction() {

$reviews = $this->model->parseReviews();

$vars = [ 'reviews' => $reviews ];

$this->view->render('GuestBookView.php', 'Гостевая книга', $vars);

}

function createAction() {

if (!empty($\_POST)) {

$this->model->validator->validate($\_POST);

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

if (empty($errors)) {

$newReview = [];

array\_push($newReview, $\_POST['fullname'], $\_POST['email'], date('Y-m-d H:i:s'), $\_POST['review']);

$this->model->addReview($newReview);

$\_POST = array();

}

$reviews = $this->model->parseReviews();

$vars = [ 'errors' => $errors, 'reviews' => $reviews ];

$this->view->render('GuestBookView.php', 'Гостевая книга', $vars);

} else {

$this->view->render('GuestBookView.php', 'Гостевая книга');

}

}

}

Модель GuestBookModel.php:

<?php

class GuestBookModel extends Model {

public function parseReviews() {

$arrStr = file("reviews.inc");

$reviews = [];

foreach (array\_reverse($arrStr) as $str) {

array\_push($reviews, preg\_split('/;/', $str));

}

return $reviews;

}

public function addReview($data) {

$file = fopen('reviews.inc', 'a+');

$str = "";

foreach($data as $item) {

$str .= $item . ';';

}

fwrite($file, "$str\n");

fclose($file);

}

}

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

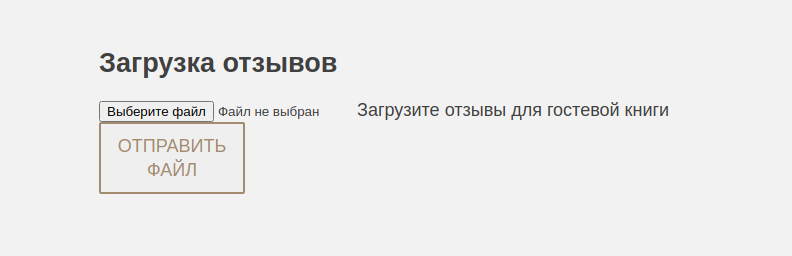


Рисунок 2 – Страница “Загрузка отзывов”

Контроллер:

<?php

class UploadReviewsController extends Controller {

function indexAction() {

$this->view->render('UploadReviewsView.php', 'Загрузка отзывов');

}

function createAction() {

if ($\_FILES['file']['name'] != "") {

if (copy ($\_FILES['file']['tmp\_name'], 'reviews.inc')) {

$vars = [ 'result' => true ];

} else {

$vars = [ 'error' => true ];

}

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

} else {

$vars = [ 'empty' => true ];

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

}

}

}

4. Разработаем страницу «Редактор Блога», позволяющую добавлять записи Блога.

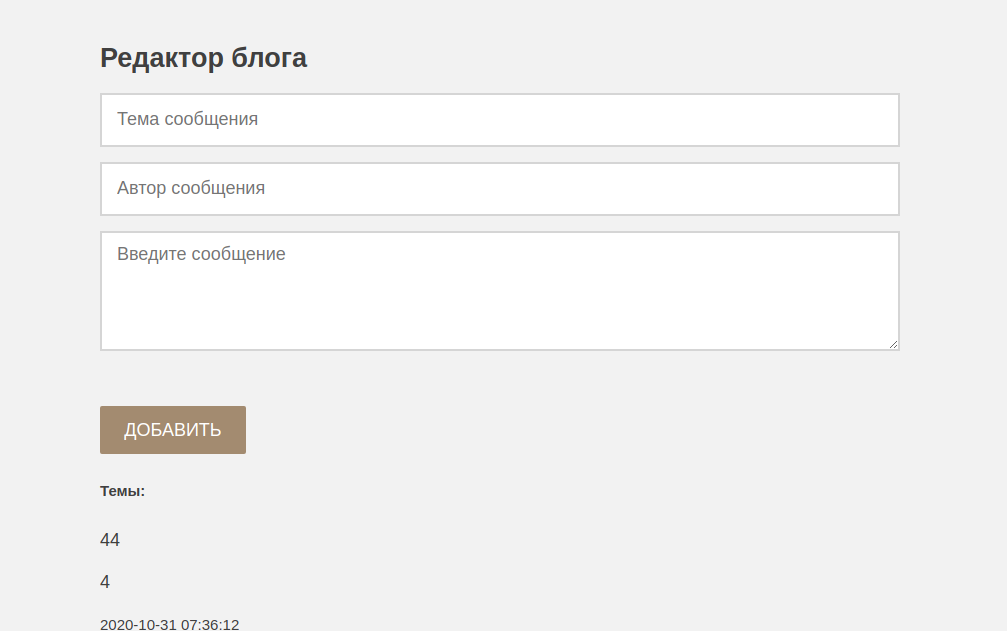


Рисунок 3 – Страница “Редактор блога”

Модели EditBlogModel.php:

<?php

require 'app/core/BaseActiveRecords.php';

class EditBlogModel extends BaseActiveRecords {

public function \_\_construct() {

parent::*\_\_construct*();

static::*$tablename* = 'blog';

static::*$dbfields* = array('title', 'author', 'text', 'date');

}

public function getPosts($get\_array) {

$countOfPosts = $this->getCount();

$rowsPerPage = 3;

$totalPages = ceil($countOfPosts / $rowsPerPage);

if (isset($get\_array['page']) && is\_numeric($get\_array['page'])) {

$currentPage = (int) $get\_array['page'];

} else {

$currentPage = 1;

}

if ((int) $currentPage > (int) $totalPages) {

$currentPage = $totalPages;

}

if ((int) $currentPage < 1) {

$currentPage = 1;

}

$offset = ($currentPage - 1) \* $rowsPerPage;

$posts = $this->findByPage($offset, $rowsPerPage);

$result = [

"posts" => $posts,

"current\_page" => $currentPage,

"total\_pages" => $totalPages

];

return $result;

}

public function addPost($post\_array) {

$data = [

"title" => $post\_array["title"],

"author" => $post\_array["author"],

"text" => $post\_array["message"]

];

array\_push($data, date('Y-m-d H:i:s'));

$this->save($data);

}

}

5. Разработаем страницу «Мой Блог», содержащую упорядоченные в порядке убывания даты добавления, посты.

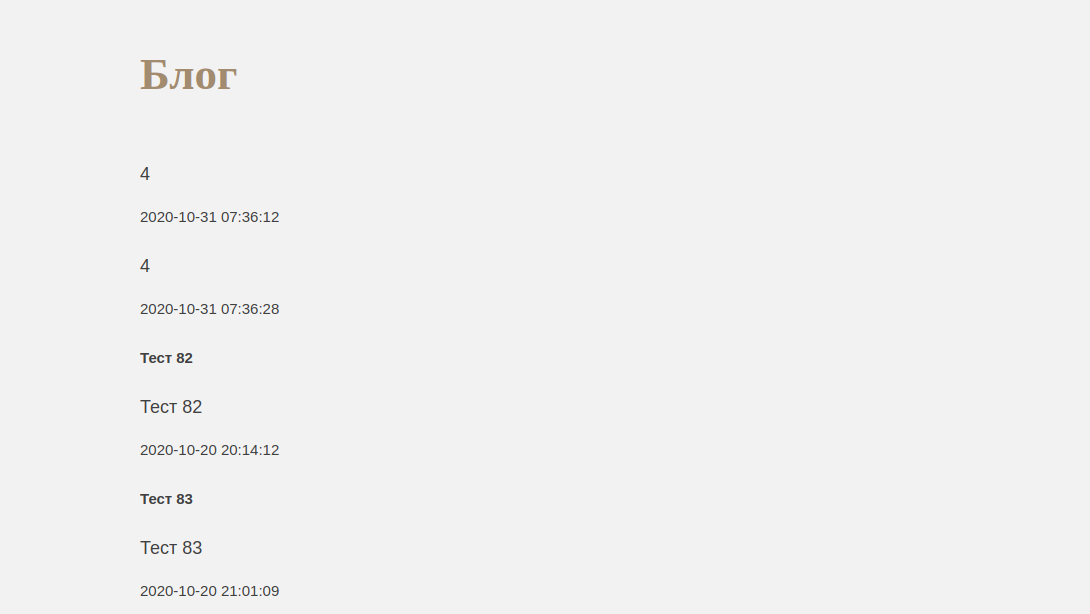


Рисунок 4 – Страница “Мой блог”

7. Реализуем возможность добавления записей на страницу «Мой Блог» из файла формата CSV. Для этого разработаем страницу «Загрузка сообщений блога», содержащую форму загрузки файла формата CSV.

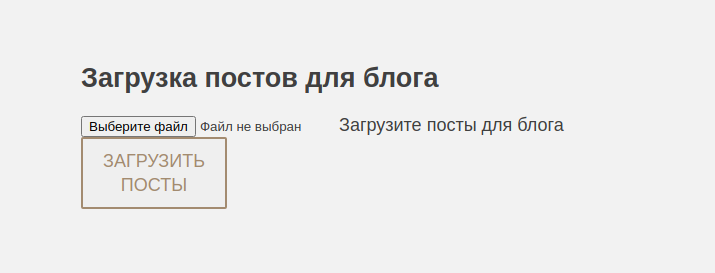


Рисунок 5 – Страница “Загрузка постов блога”

Модель страницы:

<?php

require 'app/core/BaseActiveRecords.php';

class UploadPostsModel extends BaseActiveRecords {

public function \_\_construct() {

static::*$tablename* = 'blog';

static::*$dbfields* = array('author', 'title', 'text', 'date');

}

public function savePosts($files\_array) {

$lines = file($files\_array["file"]["tmp\_name"]);

$posts = [];

foreach ($lines as $str) {

array\_push($posts, preg\_split('/;/', $str));

}

try {

foreach ($posts as $post) {

$this->save($post);

}

} catch (\Throwable $th) {

return false;

}

return true;

}

}

4. 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 овладение навыками постраничного вывода данных.