Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №3

Исследование механизма сессий в PHP

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 авторизации пользователей с использованием механизма сессий PHP, приобрести практические навыки использования переменных, сохраняющих свое значение при переходе от одной страницы сайта к другой.

2. Постановка задачи

1. В соответствии с рекомендациями реализовать зону администрирования сайта и переместить в нее страницы «Редактор блога» и «Загрузка сообщений гостевой книги».

2. Реализовать накопление информации о посетителях страниц пользовательского раздела сайта в разработанной для этого таблице базы данных.

3. Добавить в зону администрирования сайта страницу «Статистика посещений», отображающую информацию о посетителях сайта. Реализовать постраничное отображение информации в порядке убывания даты и времени посещения.

4. Добавить в зону администратора форму входа, содержащую поля логин и пароль. Логин и пароль администратора хранить к коде программы. Проверку пароля осуществлять сравнением зашифрованного пароля с вычисленным заранее хешем верного пароля.

5. На всех страницах зоны администратора реализовать проверку авторизации и в случае, если посетитель не авторизован как администратор – перенаправлять пользователя на страницу авторизации. Для этой цели возможно добавить в базовый контроллер раздела администратора метод аутентификации, который будет проверять установленную контроллером переменную сессии isAdmin.

6. В пользовательском разделе сайта разработать страницу «Регистрация пользователя», содержащую форму для ввода.

7. Добавить в меню пользовательского раздела сайта ссылку «Войти» на форму авторизации. Форма должна содержать строку ввода логина, строку ввода пароля, кнопку "Войти", а также ссылку «Регистрация» (на страницу «Регистрация пользователя»).

После успешной авторизации в меню сайта вместо ссылки «Войти» должна отображаться ссылка "Выйти", при нажатии которой происходит выход авторизированного пользователя (завершение текущей сессии).

После успешной авторизации на каждой странице сайта (возможно в области меню) должно отображаться "Пользователь: ФИО" (ФИО – Фамилия Имя Отчество пользователя).

8. Реализовать отображение результатов тестирования на странице «Тест по дисциплине» только авторизованным пользователям.

3. ход работы

1. В соответствии с рекомендациями реализуем зону администрирования сайта.

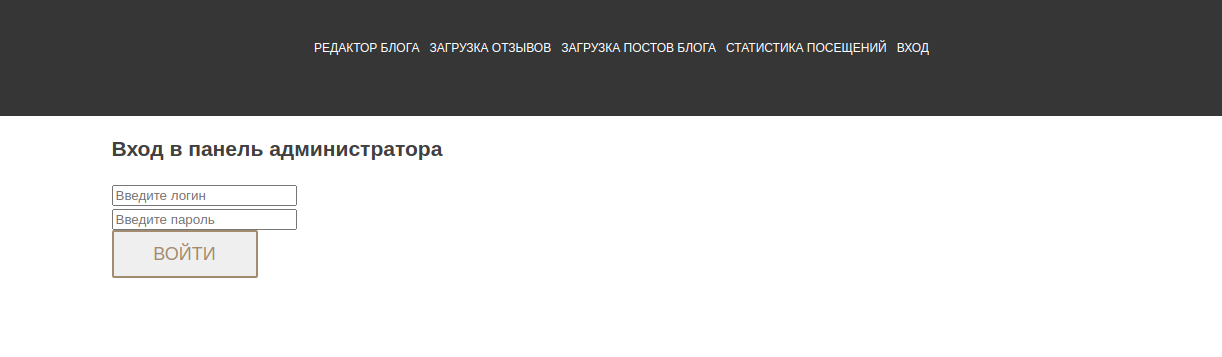


Рисунок 1 – Интерфейс зоны администрирования сайта

Для реализации зоны администратора был модифицирован класс Route следующим кодом:

<?php

class Route {

static function start() {

$controller\_name = 'Main';

$action\_name = 'index';

$admin\_prefix = "";

$admin\_path = "";

$routes = explode('/', $\_SERVER['REQUEST\_URI']);

if ($routes[1] == "admin") {

if (!empty($routes[2])) {

$controller\_name = $routes[2];

}

if (!empty($routes[3])) {

$action\_name = $routes[3];

}

$admin\_prefix = "Admin";

$admin\_path = "admin/";

} else {

if (!empty($routes[1])) {

$controller\_name = $routes[1];

}

if (!empty($routes[2])) {

$action\_name = $routes[2];

}

}

$model\_name = $admin\_prefix . $controller\_name . 'Model';

$controller\_name = $admin\_prefix . $controller\_name . 'Controller';

$action\_name = $action\_name . 'Action';

$model\_file = $model\_name . '.php';

$model\_path = "app/models/". $admin\_path . $model\_file;

if (file\_exists($model\_path)) {

include $model\_path;

}

$controller\_file = $controller\_name . '.php';

$controller\_path = "app/controllers/". $admin\_path . $controller\_file;

if (file\_exists($controller\_path)) {

include $controller\_path;

} else {

Route::*ErrorPage404*();

}

$controller = new $controller\_name;

$action = $action\_name;

if (method\_exists($controller, $action)) {

$controller->$action();

} else {

Route::*ErrorPage404*();

}

}

function ErrorPage404() {

$host = 'http://' . $\_SERVER['HTTP\_HOST'] . '/';

header('Location:' . $host.'error');

}

}

AdminLayout.php:

<!DOCTYPE html>

<html lang='en'>

<head>

<meta content='text/html; charset=UTF-8' http-equiv='Content-Type'>

<meta charset='UTF-8'>

<link href='../public/css/style.css' rel='stylesheet'>

<link href='../../public/css/style.css' rel='stylesheet'>

<link href='../public/css/lightbox.min.css' rel='stylesheet'>

<link href='../../public/css/lightbox.min.css' rel='stylesheet'>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src='../public/js/historyForEachPage.js' type='text/javascript'></script>

<script src='../../public/js/historyForEachPage.js' type='text/javascript'></script>

<title><?php echo $title; ?></title>

</head>

<body>

<div class='page-header'>

<ul class='header-nav clearfix'>

<li class="nav-item">

<a class="nav-link <?php if ($title == "Редактор блога") echo('menu-link-active') ?>" href="/admin/EditBlog">Редактор блога</a>

</li>

<li class="nav-item">

<a class="nav-link <?php if ($title == "Загрузка отзывов") echo('menu-link-active') ?>" href="/admin/UploadReviews">Загрузка отзывов</a>

</li>

<li class="nav-item">

<a class="nav-link <?php if ($title == "Загрузка постов блога") echo('menu-link-active') ?>" href="/admin/UploadPosts">Загрузка постов блога</a>

</li>

<li class="nav-item">

<a class="nav-link <?php if ($title == "Статистика посещений") echo('menu-link-active') ?>" href="/admin/Statistics">Статистика посещений</a>

</li>

<li class="nav-item">

<?php

if (isset($\_SESSION['isAdmin'])) {

?>

<a class="nav-link" href="/admin/Auth/logout">Выход</a>

<?php

} else {

?>

<a class="nav-link <?php if ($title == "Вход") echo('menu-link-active') ?>" href="/admin/Auth">Вход</a>

<?php

}

?>

</li>

</ul>

</div>

<?php

include 'app/views/admin/' . $content\_view;

?>

</body>

</html>

2. Реализуем накопление информации о посетителях страниц пользовательского раздела сайта в разработанной для этого таблице базы данных и добавим в зону администрирования сайта страницу «Статистика посещений», отображающую информацию о посетителях сайта. Реализуем постраничное отображение информации в порядке убывания даты и времени посещения.

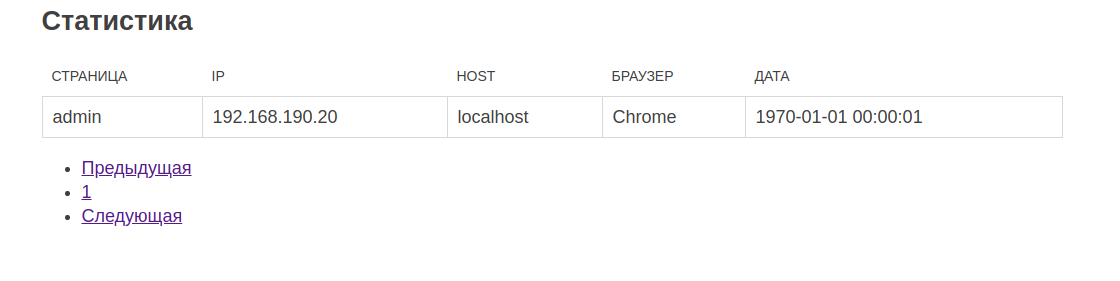


Рисунок 2 – Страница просмотра статистики посещений

Для сохранения информации о посещении каждой страницы был модифицирован базовый класс Model следующим методом:

public function saveUserInfo($page) {

static::setupConnection();

$tablename = 'statistics';

$sql = "

INSERT INTO $tablename (`page`, `ip`, `host`, `browser`, `date`)

VALUES (

'" . $page . "',

'" . $\_SERVER['REMOTE\_ADDR'] . "',

'" . gethostbyaddr($\_SERVER['REMOTE\_ADDR']) . "',

'" . $\_SERVER['HTTP\_USER\_AGENT'] . "',

'" . date('Y-m-d H:i:s') . "'

)

";

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

}

Контроллер AdminStatisticsController:

class AdminStatisticsController extends AdminController {

function \_\_construct() {

*$this*->model = new AdminStatisticsModel();

*$this*->view = new View();

}

function indexAction() {

*$this*->checkIsAuth();

$result = *$this*->model->getStatistics($\_GET);

*$this*->view->render('AdminStatisticsView.php', 'Статистика посещений', $result, 'AdminLayoutView.php');

}

}

Модель AdminStatisticsModel:

class AdminStatisticsModel extends Model {

public function \_\_construct() {

parent::\_\_construct();

static::$tablename = 'statistics';

static::$dbfields = array('page', 'ip', 'host', 'browser', 'date');

}

public function getStatistics($get\_array) {

$countOfPosts = *$this*->getCount();

$rowsPerPage = 6;

$totalPages = ceil($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;

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

$result = [

"statistics" => $statistics,

"current\_page" => $currentPage,

"total\_pages" => $totalPages

];

*return* $result;

}

}

3. Добавим в зону администратора форму входа, содержащую поля логин и пароль. Логин и пароль администратора будем хранить в коде программы. Проверку пароля будем осуществлять сравнением зашифрованного пароля с вычисленным заранее хешем верного пароля.

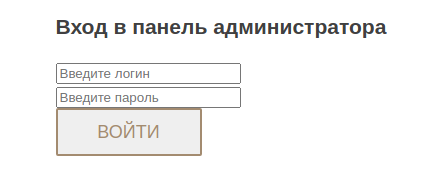


Рисунок 3 – Форма входа в зону администратора

Для реализации входа в зону администратора был реализован контроллер AdminAuthController:

class AdminAuthController extends Controller {

function indexAction() {

*$this*->view->render('AdminAuthView.php', ] 'Вход', null, 'AdminLayoutView.php');

}

function loginAction() {

*if* (!empty($\_POST)) {

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

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

*if* (empty($errors)) {

*if* (*$this*->compareLoginData($\_POST)) {

header('Location:/admin/editblog');

*exit*;

} *else* {

$login = false;

$\_POST['password'] = null;

}

}

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

*$this*->view->render('AdminAuthView.php', 'Вход', $vars, 'AdminLayoutView.php');

} *else* {

*$this*->view->render('AdminAuthView.php', 'Вход', null, 'AdminLayoutView.php');

}

}

function logoutAction() {

unset($\_SESSION['isAdmin']);

header('Location:/admin/auth');

*exit*;

}

function compareLoginData($post\_array) {

*if* (($post\_array['login'] === 'admin') && ($post\_array['password'] === 'admin')) {

$\_SESSION['isAdmin'] = 1;

*return* true;

}

*return* false;

}

}

4. На всех страницах зоны администратора реализуем проверку авторизации и в случае, если посетитель не авторизован как администратор – будем перенаправлять пользователя на страницу авторизации. Для этой цели добавим в базовый контроллер раздела администратора метод аутентификации, который будет проверять установленную контроллером переменную сессии isAdmin.

Базовый контроллер раздела администратора AdminController:

class AdminController extends Controller {

function checkIsAuth() {

*if* (!isset($\_SESSION['isAdmin'])) {

header('Location:/admin/auth');

*exit*;

}

}

}

5. В пользовательском разделе сайта разработаем страницу «Регистрация пользователя», содержащую форму для ввода.

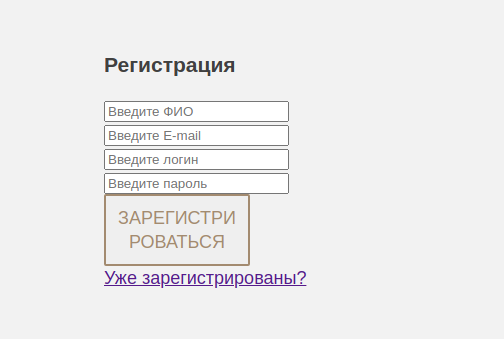


Рисунок 4 – Форма регистрации

Класс AuthController:

class AuthController extends Controller {

function \_\_construct() {

*$this*->model = new AuthModel();

*$this*->view = new View();

}

function signinAction() {

*$this*->view->render('SignInView.php', 'Вход', null, 'LayoutView.php');

}

function signupAction() {

*$this*->view->render('SignUpView.php', 'Регистрация', null, 'LayoutView.php');

}

function loginAction() {

*if* (!empty($\_POST)) {

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

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

*if* (empty($errors)) {

$findUser = *$this*->model->findUser($\_POST);

*if* ($findUser) {

$\_SESSION['isUser'] = 1;

$\_SESSION['userFullname'] = $findUser->fullname;

header('Location:/');

*exit*;

} *else* {

$login = false;

$\_POST['password'] = null;

}

}

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

*$this*->view->render('SignInView.php', 'Вход', $vars, 'LayoutView.php');

} *else* {

*$this*->view->render('SignInView.php', 'Вход', null, 'LayoutView.php');

}

}

function createAction() {

*if* (!empty($\_POST)) {

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

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

*if* (empty($errors)) {

*if* (*$this*->model->createUser($\_POST)) {

$\_POST = array();

header('Location:/auth/signin');

*exit*;

} *else* {

$result = false;

}

}

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

*$this*->view->render('SignUpView.php', 'Регистрация', $vars, 'LayoutView.php');

} *else* {

*$this*->view->render('SignUpView.php', 'Регистрация', null, 'LayoutView.php');

}

}

function logoutAction() {

unset($\_SESSION['isUser']);

header('Location:/auth/signin');

*exit*;

}

}

Модель AuthModel:

class AuthModel extends Model {

public function \_\_construct() {

parent::\_\_construct();

static::$tablename = 'users';

static::$dbfields = array('fullname', 'email', 'login', 'password');

}

public function createUser($post\_array) {

$findUserByEmail = *$this*->findByField($post\_array['Email'], 'email');

$findUserByLogin = *$this*->findByField($post\_array['login'], 'login');

*if* ($findUserByEmail != null || $findUserByLogin != null) {

*return* false;

}

$data = [

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

"email" => $post\_array['Email'],

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

"password" => $post\_array["password"]

];

*$this*->save($data);

*return* true;

}

public function findUser($post\_array) {

$user = *$this*->findByQuery("`login`='".$post\_array['login']."' AND `password`='".$post\_array['password']."'");

*if* ($user == null) {

*return* false;

}

*return* $user;

}

}

6. Добавим в меню пользовательского раздела сайта ссылку «Войти» на форму авторизации. Форма должна содержать строку ввода логина, строку ввода пароля, кнопку "Войти", а также ссылку «Регистрация» (на страницу «Регистрация пользователя»).

После успешной авторизации в меню сайта вместо ссылки «Войти» должна отображаться ссылка "Выйти", при нажатии которой происходит выход авторизированного пользователя (завершение текущей сессии).

После успешной авторизации на каждой странице сайта (возможно в области меню) должно отображаться "Пользователь: ФИО" (ФИО – Фамилия Имя Отчество пользователя).

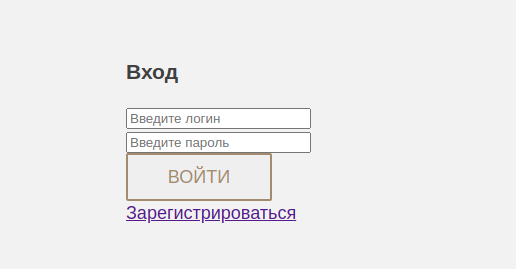


Рисунок 5 – Форма входа

Рисунок 6 – Отображение ФИО залогиненного пользователя

Весь код авторизации и регистрации представлен в 5 пункте хода работы.

4. Вывод

В ходе выполнения лабораторной работы были изучены возможности авторизации пользователей с использованием механизма сессий PHP, приобретены практические навыки использования переменных, сохраняющих свое значение при переходе от одной страницы сайта к другой.