Министерство науки и высшего образования Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: «Веб-технологии»

Лабораторная работа № 3

«Исследование механизма сессий в PHP»

Выполнила

ст. гр. ИС/б-17-2-о

Кайда Е.Н.

Проверил:

Овчинников А.Л.

Севастополь

2020

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.

Постановка задачи

1. 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. Структура информации:

* Дата и время посещения;
* Web-страница посещения;
* IP-адрес компьютера посетителя;
* Имя хоста компьютера посетителя;
* Название браузера, который использовал посетитель.

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 админстратора 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 (все поля обязательны к заполнению):

* ФИО пользователя;
* e-mail пользователя;
* логин пользователя;
* пароль пользователя.

Добавлять введенную пользователем информацию в разработанную таблицу базы данных. Предусмотреть проверку на невозможность добавления двух одинаковых логинов.

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.

ход работы

1. Реализовать зону администрирования сайта и переместить в нее страницы «Редактор блога» и «Загрузка сообщений гостевой книги». Для этого создадим папку admin в app и добавим проверку на существование параметра admin\_area в url:

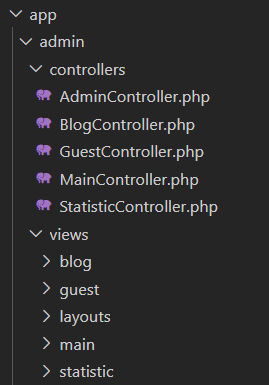


Рисунок 1 – Создание директории admin

**.htaccess**

RewriteRule ^admin(.\*)$ index.php?admin\_area=1 [L]

**Router.php и View.php**

**//если это зона администратора, то изменить путь в папку админ**

if (array\_key\_exists('admin\_area', $\_REQUEST)){

            $admin\_path = "admin\\";

}

1. Реализовать накопление информации о посетителях страниц пользовательского раздела сайта в разработанной для этого таблице базы данных. Для этого был создан класс Statistic:

class Statistic extends BaseActiveRecordModel{

    public static $tablename = 'statistics';

    public static $fields = [];

    public static $pdo;

    public $date;

    public $web\_page;

    public $ip\_address;

    public $host\_name;

    public $browser\_name;

    public function save\_statistic($page){

        // echo "<p style='margin-left: 50px;'> save\_statistic </p>";

        $this->date = date('Y-m-d h:m:s');

        $this->web\_page = $page;

        $this->ip\_address = $\_SERVER['REMOTE\_ADDR'];

        $this->host\_name = gethostbyaddr($\_SERVER['REMOTE\_ADDR']);

        $this->browser\_name = $\_SERVER['HTTP\_USER\_AGENT'];

        $this->save([[null, $this->date, $this->web\_page, $this->ip\_address,

                        $this->host\_name, $this->browser\_name]]);

    }

}

В основном контроллере при создании вызывается метод *save\_statistic:*

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

            $this->stat = new Statistic($this->route['action']);

            $this->stat->save\_statistic($this->route['controller'].'/'.$this->route['action']);

}

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

Контроллер класса Статистики из зоны администратора:

class StatisticController extends AdminController{

    public function  visitingsAction(){

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

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

    }

}

Добавление функционала в модель:

function visitings(){

        $args = $this->linksPages();

        $args['data'] = parent::findAll();

        return $args;

}

Представление данной страницы *visitings.php:*

<section>

    <h1> Статистика посещаемости </h1>

    <table class="comments">

        <tbody>

            <?php

                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 "<h4>" .$temp['web\_page']."</h4></td>";

                    echo "<td class=\"text\"><p>IP: ".$temp['ip']."</p>";

                    echo "<p>Host: ".$temp['host']."</p>";

                    echo "<p>Browser: ".$temp['browser']."</p>";

                    echo "</td></tr>";

                    // echo "<br/>".var\_dump($temp);

                }

            ?>

        </tbody>

    </table>

    <br/>

    <?php

     foreach($pages as $i){

          echo $i;

     }

     ?>

</section>

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

Форма входа в зону администратора:

<?php

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

                $controller->login();

                header('Location: /admin');

        }

?>

<section>

        <h1>Админ панель</h1>

        <br/>

        <?php if (!isset($\_SESSION['isAdmin']) || !$\_SESSION['isAdmin']): ?>

        <form class="adminForm" action="" method="POST">

                <h2>Авторизация</h2>

                <p><input type="text" name="login" placeholder="Login"></p>

                <p><input type="password" name="password" placeholder="Password"></p>

                <p><input type="submit" name="submit" value="Войти"></p>

        </form>

        <?php else: ?>

        <h2> Вы зашли под аккаунтом администратора.</h2>

        <p> Здесь вы можете изменять данные web-страниц.</p>

        <?php endif ?>

</section>

Контроллер:

class MainController extends Controller{

    public function \_\_construct($route){

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

    }

     public function  indexAction(){

         $var = [

             'controller' => $this,

         ];

         $this->view->render('Admin', $var);

     }

     public static function login(){

        if (($\_POST['login']=='adminadmin@gmail.com') && (md5($\_POST['password'])=='21232f297a57a5a743894a0e4a801fc3')) { //admin

//Сохранение переменной 'isAdmin' в сессии

            $\_SESSION['isAdmin']=1;

            return true;

        }

        return false;

     }

}

1. На всех страницах зоны администратора реализовать проверку авторизации и в случае, если посетитель не авторизован как администратор –перенаправлять пользователя на страницу авторизации.

Для этого в зоне администратора был создан AdminController с функцией проверки пользователя на администратора и от этого контроллера были унаследованы все контроллеры в директории admin:

class AdminController extends Controller{

    public function \_\_construct($route){

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

        if (isset($\_SESSION['isAdmin']) && $\_SESSION['isAdmin']) echo "<br/> is admin";

        $this->authenticate();

    }

    public static function authenticate(){

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

            header('Location:/admin');

            exit;

        }

    }

}

class BlogController extends AdminController

class GuestController extends AdminController

class StatisticController extends AdminController

1. В пользовательском разделе сайта разработать страницу «Регистрация пользователя», содержащую форму для ввода. Добавлять введенную пользователем информацию в разработанную таблицу базы данных. Предусмотреть проверку на невозможность добавления двух одинаковых логинов.

Представление формы регистрации пользователя:

<section>

<?php

        $values;

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

                if (!$controller->registration()):

                        $values = $\_POST;

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

                else:

                        $values = [];

                        echo "<h1 style='color:darkgreen;'> Регистрация прошла успешно </h1><br/>";

                endif;

        endif;

?>

        <h1>Регистрация пользователя</h1>

        <br/>

        <form action="" method="POST">

            <p><input type="text" name="name" placeholder="ФИО" data-toggle="popover"

                value="<?php if (isset($values['name'])) echo $values['name'] ?>" data-content="<?php echo $rules['name'] ?>"><br><pre><?php if (isset($errors['name'])) echo $errors['name'] ?><br></pre><br></p>

            <p><input type="text" name="email" placeholder="Email" data-toggle="popover"

                value="<?php if (isset($values['email'])) echo $values['email'] ?>" data-content="<?php echo $rules['email'] ?>"><br><pre><?php if (isset($errors['email'])) echo $errors['email'] ?><br></pre><br></p>

            <p><input type="text" name="login" placeholder="Логин" data-toggle="popover"

                value="<?php if (isset($values['login'])) echo $values['login'] ?>" data-content="<?php echo $rules['login'] ?>"><br><pre><?php if (isset($errors['login'])) echo $errors['login'] ?><br></pre><br></p>

            <p><input type="password" name="password" placeholder="Пароль" data-toggle="popover"

                value="<?php if (isset($values['password'])) echo $values['password'] ?>" data-content="<?php echo $rules['password'] ?>"><br><pre><?php if (isset($errors['password'])) echo $errors['password'] ?><br></pre><br></p>

            <p><input type="submit" name="submit" value="Зарегестрироваться"></p>

        </form>

</section>

Контроллер:

class UserController extends Controller{

    public function registrationAction(){

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

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

    }

    public function loginAction(){

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

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

    }

//Метод проверки зарегестрирован ли пользователь, если да, то произвести вход

    public function login(){

        if (MainController::login()) {

            header('Location: /admin');

            return true;

        }

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

        if (!$this->model->validator->checkErrors()) return false;

        $this->setUserSession();

        $this->redir();

    }

//Метод регистрации пользователя с проверкой на существование такого логина

    public function registration(){

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

        $this->data['errors'] = $this->model->validator->getErrors();

        if (!$this->model->validator->checkErrors()) return false;

        $this->model->saveUser();

        return true;

    }

//Инициализация пользователя в сессии

    public function setUserSession(){

        $\_SESSION['isAdmin'] = 0;

        $\_SESSION['fio'] = $this->model->fio;

        $this->redir();

    }

    public function redir(){

        header('Location: /');

    }

}

Модель:

class User extends BaseActiveRecordModel{

    protected static  $tablename = "users";

    protected static $dbfields = [];

    public static  $pdo;

    public $fio;

    public function \_\_construct($action){

        //  echo "<p>Blog Model \_\_construct(action)</p>";

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

    }

    public function registration(){

        $this->validator = new ValidRegistration($this);

        return[

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

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

        ];

    }

    public function login(){

        $this->validator = new ValidLogin($this);

        return[

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

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

        ];

    }

    public function exit(){}

    public function check\_exisists\_login($login){

        $search = $this->find("`login`", "'".$login."'");

        return empty($search);      //true если пусто

    }

    public function check\_exisists\_login\_password($login, $password){

        $sql = "SELECT \* FROM ".static::$tablename." WHERE `login`='$login' AND `password`='$password'";

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

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

        var\_dump($stmt);

        if ($stmt == false) return false;

        $this->fio = $stmt['name'];

        return true;

    }

    public function saveUser(){

        $val = [

            [null, $\_POST['name'], $\_POST['email'], $\_POST['login'], $\_POST['password'] ]

        ];

        $this->save($val);

    }

}

1. Добавить в меню пользовательского раздела сайта ссылку «Войти» на форму авторизации. После успешной авторизации в меню сайта вместо ссылки «Войти» должна отображаться ссылка "Выйти. После успешной авторизации на каждой странице сайта должно отображаться "Пользователь: ФИО" (ФИО – Фамилия Имя Отчество пользователя).

Для реализации данной задачи было созданно представление авторизации:

<?php

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

                if (!$controller->login()):

                        $values = $\_POST;

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

                endif;

        endif;

?>

<section>

        <br/>

        <h1>Авторизация</h1>

        <br/>

        <form class="adminForm" action="" method="POST">

                <p><input type="text" name="login" placeholder="Логин" data-toggle="popover"

                        value="<?php if (isset($values['login'])) echo $values['login'] ?>" data-content="<?php echo $rules['login'] ?>"><br><pre><?php if (isset($errors['login'])) echo $errors['login'] ?><br></pre><br></p>

                <p><input type="password" name="password" placeholder="Пароль" data-toggle="popover"

                        value="<?php if (isset($values['password'])) echo $values['password'] ?>" data-content="<?php echo $rules['password'] ?>"><br><pre><?php if (isset($errors['password'])) echo $errors['password'] ?><br></pre><br></p>

                <p>

                        <input type="submit" value="Войти" style="width: 160px;">

                        <a href="/registration"><input type='button' id="registration" value="Зарегестрироваться" style="width: 160px;"></a>

                </p>

        </form>

</section>

Модель User и контроллер UserController представлены выше. При верной авторизации в контроллере вызывается метод setUserSession() который сохраняет фио пользователя и перенаправляет нас на главную страницу.

Так же был изменен файл с выводом меню:

<body>

<input type="checkbox" id="nav-toggle" hidden>

    <nav class="nav">

        <label for="nav-toggle" class="nav-toggle" onclick></label>

        <h2 class="logo">

            <a>Little\_coon\_</a>

        </h2>

        <?php

//если пользователь авторизирован как администратор, то выводить дополнительные пункты меню

            if(isset($\_SESSION['isAdmin']) && $\_SESSION['isAdmin']):

                echo "<h2>Admin</h2>";

                echo "<h3><a href='/exit'>Выход</a></h3>";

        ?>

        <ul>

            <li><a href="/admin/guest/upload">Загрузка отзывов</a>

            <li><a href="/admin/blog/upload">Загрузка сообщений блога</a>

            <li><a href="/admin/statistic/visitings">Статистика посещаемости</a>

            <?php

                else:

                    echo "<ul>";

//если пользователь авторизирован

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

                        echo "<h2>".$\_SESSION['fio']."</h2>";

                        echo "<h3><a href='/exit'>Выход</a></h3>";

                    }else{

//если пользователь не авторизирован

                        echo "<h3><a href='/login'>Вход</a></h3>";

                    }

                endif;

            ?>

            <li><a href="/">Главная</a>

            <li><a href="/aboutMe">Обо мне</a>

            <li><a href="/interests">Мои интересы</a>

            <li><a href="/schedule">Учеба</a>

            <li><a href="/fotos">Фотоальбом</a>

            <li><a href="/test">Тест</a>

            <li><a href="/contacts">Контакты</a>

            <li><a href="/guestBook">Гостевая книга</a>

            <li><a href="/blog">Мой Блог</a>

            <li><a href="/blogEditor">Редактор Блога</a>

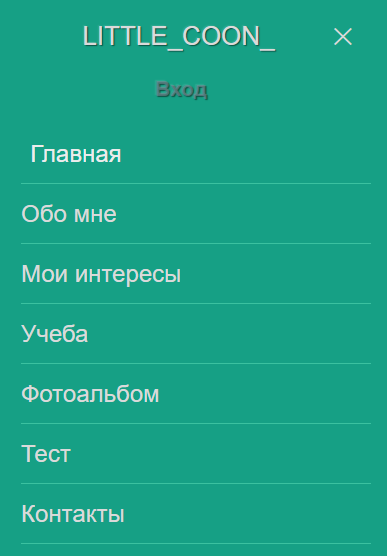
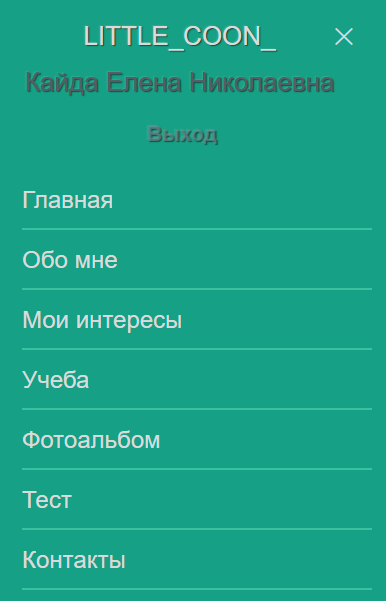
            <li><a href="/registration">Регистрация</a>

        </ul>

    </nav>

    <?php require $view; ?>

</body>

А Б

Рисунок 2 – Вид меню сайта если пользователь А) не авторизирован, Б) авторизирован

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

Для этого в представление из модели посылается еще одна переменная user? Которая означает авторизирован пользователь или нет:

'user' => (isset($\_SESSION['fio']) || $\_SESSION['isAdmin'])

А в представление производится следующая обработка:

<?php if(!$user):

            echo "<h2>Авторизируйтесь для просмотра результатов тестирования</h2>";

        else:

            if(empty($results)):

                echo "<h2>Нет результатов</h2>";

            else:

?>

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

Выводы

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.

Приложение А

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