МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федерально автономное образовательное учреждение высшего образования
«Севастопольский государственный университет»
кафедра Информационных систем

Куркчи Ариф Эрнестович

Институт информационных технологий и управления в технических системах
курс 3 группа ИС/б-31-о
09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №7

по дисциплине «Веб-технологии»

на тему «Исследование механизма сессий в PHP.»

Отметка о зачете _____ _____
(дата)

Руководитель практикума

ст. преподаватель	_____	А. Л. Овчинников
(должность)	(подпись)	(инициалы, фамилия)

Севастополь 2017

1. ЦЕЛЬ РАБОТЫ

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

2. ПОСТАНОВКА ЗАДАЧИ

2.1. Реализовать зону администрирования сайта. Для этого:

- в корневой директории Web-сайта создать папку «admin»;
- создать главную страницу раздела администратора(admin\index.php), имеющую фреймовую структуру, и содержащую два фрейма – меню и содержимое;
- переместить в зону администрирования страницы «Редактор блога», «Загрузка сообщений гостевой книги»;
- на всех страницах зоны администратора реализовать проверку авторизации администратора. В случае, если посетитель не авторизован как администратор, отображать форму ввода логина и пароля для авторизации. Логин и пароль администратора должен храниться в файле (например, admin\pswd.inc);

2.2. 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.

2.3. Добавить в зону администрирования сайта страницу «Статистика посещений», отображающую информацию о посетителях сайта (п. 3.2.). Реализовать постраничное отображение информации в порядке убывания даты и времени посещения.

2.4. 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.

2.5. Модифицировать главную страницу пользовательского раздела сайта, добавив форму авторизации. Форма должна содержать строку ввода логина, строку ввода пароля, кнопку "Войти", а также ссылку «Регистрация» (на страницу «Регистрация пользователя»).

После успешной авторизации на главной странице сайта вместо формы авторизации должна появиться кнопка(или ссылка) "Выйти", при нажатии которой происходит выход авторизированного пользователя (завершение сессии).

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

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

3. ИСХОДНЫЙ КОД

app/Controller/UserController

```php
<?php

namespace App\Controller;

use App\Core\Auth;
use App\Core\Controller;
use App\Core\Form;
use App\Model\User;
use App\Validation\UserVerification;
use function App\Core\view;

class UserController extends Controller {

    public $_authOnly = [ 'stats' ];
    protected $_rulesSignUp = [
        'name'      => 'required,word:1',
        'password'  => 'required,min:8',
        'email'     => 'required,email',
        'real_name' => 'required,words:2-3',
    ];
    protected $_rulesSignIn = [
        'login'    => 'required',
        'password' => 'required',
    ];
```

```
    protected $_messages = [
        'login'     => [
            'sign_in' => 'Логин или пароль не правильный',
        ],
        'password'  => [
            'min' => 'Пароль должен быть не меньше 8 символов',
        ],
        'name'      => [
            'required' => 'Поле обязательное',
            'sign_up'  => 'Пользователь с таким логином уже зарегистрирован',
        ],
        'real_name' => [
            'required' => 'Поле обязательное',
            'words'    => 'Введите в формате: Фамилия Имя Отчество',
        ],
        'email'     => [
            'required' => 'Поле обязательное',
            'email'    => 'Введите действительную почту',
            'sign_up'  => 'Пользователь с такой почтой уже зарегистрирован',
        ],
        'tel'       => [
            'required' => 'Поле обязательное',
            'tel'      => 'Допустимы действительные номера +7 и +380',
        ],
        'sex'       => 'Поле обязательное',
        'age'       => 'Поле обязательное',
        'dob'       => [
            'required' => 'Поле обязательное',
            'date'     => 'Введите в формате: день.месяц.год',
        ],
    ];

    function sign_in() {
        if ( ! Auth::instance()->is_guest ) {
            header( 'Location: /' );
            exit;
        }
        $form = new Form( $_REQUEST, UserVerification::run( $this->_rulesSignIn, $this->_messages, 'sign_in' ) );
        if ( $form->success() ) {
            $_SESSION['login']    = trim( $form->val( 'login' ) );
            $_SESSION['password'] = $form->val( 'password' );
            session_commit();
            header( 'Location: /' );
            exit;
        }

        view( 'user.sign_in', compact( 'form' ) );
    }

    function sign_up() {
        if ( ! Auth::instance()->is_guest ) {
            header( 'Location: /' );
            exit;
        }
        $form = new Form( $_REQUEST, UserVerification::run( $this->_rulesSignUp, $this->_messages, 'sign_up' ) );
        if ( $form->success() ) {
            User::create( [
                'name'      => trim( $form->val( 'name' ) ),
                'password'  => password_hash( $form->val( 'password' ), PASSWORD_BCRYPT ),
                'email'     => trim( $form->val( 'email' ) ),
                'real_name' => $form->val( 'real_name' ),
                'role'      => 'user',
            ] );
            $_SESSION['login']    = trim( $form->val( 'name' ) );
            $_SESSION['password'] = $form->val( 'password' );
            session_commit();
            header( 'Location: /' );
            exit;
        }

        view( 'user.sign_up', compact( 'form' ) );
    }

    function sign_out() {
        Auth::instance()->logout();
        header( 'Location: /' );
        exit;
    }

}
```

app/Validation/UserValidation

```
<?php

namespace App\Validation;

use App\Core\Validation;
use App\Model\User;

class UserVerification extends Validation {

    protected $_method;

```

```
    public static function run( array $rules, array $messages = [], $method = 'sign_in' ) {
        $validator          = new static( $_REQUEST, $rules, $messages );
        $validator->_method = $method;
        if ( ! empty( $validator->_form ) ) {
            $validator->validate();
        }

        return $validator;
    }

    public function validate() {
        parent::validate();

        if ( $this->_method == 'sign_up' && ! empty( $this->_errors ) ) {
            $this->verify();
        }

        return $this;
    }

    public function verify() {
        switch ( $this->_method ) {
            case 'sign_in':
                $this->verifySignIn();
                break;
            case 'sign_up':
                $this->verifySignUp();
                break;
        }
    }

    public function verifySignIn() {
        $user = User::auth( $this->_form['login'], $this->_form['password'] );
        if ( $user !== false ) {
            $this->_result = 'Вы успешно авторизованы';
        } else {
            $this->_errors['login'] = $this->errorMessage( 'login', 'sign_in' );
        }
    }

    public function verifySignUp() {
        $by_name  = User::findBy( 'name', $this->_form['name'], true );
        $by_email = User::findBy( 'email', $this->_form['email'], true );

        if ( $by_name !== false ) {
            $this->_errors['name'] = $this->errorMessage( 'name', 'sign_up' );
        }
        if ( $by_email !== false ) {
            $this->_errors['email'] = $this->errorMessage( 'email', 'sign_up' );
        }
        if ( empty( $this->_errors ) ) {
            $this->_result = 'Вы успешно зарегистрированы';
        }
    }

}
```

app/Model/Auth

```
<?php

namespace App\Model;

use App\Core\ActiveRecord;

class User extends ActiveRecord {

    protected static $table = 'users';
    protected static $fields = [
        'id',
        'name',
        'password',
        'email',
        'real_name',
        'role',
        'created_at',
        'updated_at',
    ];

    public static function auth($login, $password) {
        $searchField = preg_match('/.+@.+\..+/', $login) ? 'email' : 'name';
        $user = static::findBy($searchField, $login, true);
        if($user !== false && password_verify($password, $user->password)) {
            return $user;
        }
        return false;
    }

}
```

app/Core/Router

```php
<?php

namespace App\Core;

use App\Model\Visit;

class Router {
    private static $route = [];

    public static function run() {
        define( '_AJAX', ! empty( $_SERVER['HTTP_X_REQUESTED_WITH'] ) && strtolower( $_SERVER['HTTP_X_REQUESTED_WITH'] ) == 'xmlhttprequest' );

        $uri = $_SERVER['REQUEST_URI'];
        $len = 0;
        if ( ! empty( $uri ) ) {
            foreach ( explode( '/', $uri ) as $item ) {
                if ( ! empty( $item ) ) {
                    self::$route[] = $item;
                    $len ++;
                }
            }
        }

        if ( $len < 1 ) {
            self::$route[] = 'home';
            $len ++;
        }
        if ( $len < 2 ) {
            self::$route[] = 'index';
            $len ++;
        }

        $className  = sprintf( '\\App\\Controller\\%sController', ucfirst( strtolower( self::$route[0] ) ) );
        $methodName = strtolower( self::$route[1] );
        $args       = array_slice( self::$route, 2, $len - 2 );
        Visit::run(self::$route[0], self::$route[1]);
        View::share( [
            '_controller' => strtolower( self::$route[0] ),
            '_method'     => strtolower( self::$route[1] ),
            '_args'       => $args,
            '_route'      => implode( '/', self::$route ),
            '_href'       => '/' . ( self::$route[0] == 'home' ? '' : self::$route[0] ) . ( $methodName == 'index' ? '' : '/' . $methodName ),
        ] );
        try {
            $reflectionClass  = new \ReflectionClass( $className );
            $reflectionMethod = $reflectionClass->getMethod( $methodName );
            $obj              = null;
            if ( ! $reflectionMethod->isStatic() ) {
                $obj = $reflectionClass->newInstance();
                View::share( '_controller', $obj );
            }
            if ( Auth::instance()->is_guest && $reflectionClass->hasProperty( '_authOnly' ) && in_array( $methodName, $reflectionClass->getProperty( '_authOnly' )->getValue( $obj ) ) ) {
                Error::_403( 'Auth method only' );
            }
            if ( ! Auth::instance()->is_admin && $reflectionClass->hasProperty( '_adminOnly' ) && in_array( $methodName, $reflectionClass->getProperty( '_adminOnly' )->getValue( $obj ) ) ) {
                Error::_403( 'Admin method only' );
            }
            if ( ! _AJAX && $reflectionClass->hasProperty( '_ajaxOnly' ) && in_array( $methodName, $reflectionClass->getProperty( '_ajaxOnly' )->getValue( $obj ) ) ) {
                Error::_405( 'AJAX method only' );
            }
            $reflectionMethod->invokeArgs( $obj, $args );
        } catch ( \ReflectionException $e ) {
            Error::_404( $e->getMessage() );
        }
    }
}
```

Вывод

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