МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Гусев Владислав Яковлевич

Институт информационных технологий и управления в технических системах

курс 3 группа ГИС/б-32-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №4

по дисциплине: «Веб-технологии»

по теме: «Исследование возможностей асинхронного взаимодействия с сервером. Технология AJAX»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. пр. Забаштанский А.К.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2019

1. ЦЕЛЬ РАБОТЫ

Изучить возможности асинхронного обмена данными с сервером, приобрести практические навыки использования технологии AJAX для организации отправки данных на сервер и получения данных без полной перезагрузки страницы.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. На странице «Регистрация пользователя» реализовать проверку занятости логина без перезагрузки страницы (использовать событие onBlur или кнопку «Проверить занятость»).

2. Реализовать возможность комментирования записей блога для авторизированных пользователей сайта. В случае, если пользователь авторизирован отображать после каждой записи блога кнопку или ссылку «Добавить комментарий». При нажатии кнопки/ссылки отображать окно (div), содержащее поле ввода текста комментария и кнопку «Отправить». Реалиовать отправку комментария на сервер и его отображение (дата и время, автор, текст комментария) под записью блога (или под уже существующими комментариями данной записи). Комментарии хранить в специально разработанной таблице базы данных. Использовать для отправки данных на сервер метод и формат данных, указанный в колонке «Задание №2» таблицы 4.3 в соответствии с вариантом задания.

3. В зоне администратора реализовать возможность редактирования записей блога. Отображать возле каждой записи блога кнопку или ссылку «Изменить». При нажатии кнопки/ссылки отображать окно (div), содержащее поле ввода с темой записи блога и поле ввода с текстом записи бло-га для редактирования, а также кнопку/ссылку «Сохранить изменения». Использовать для отправки данных на сервер метод и формат данных, указанный в колонке «Задание №3» таблицы 4.3 в соответствии с вариантом задания.

1. ХОД РАБОТЫ

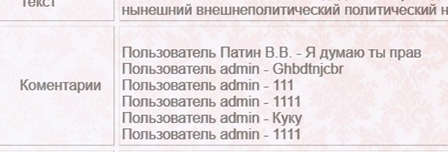


Рисунок 1 – Комментарии к записи

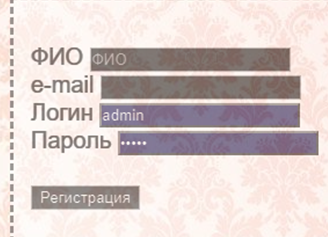


Рисунок 2 – Форма регистрации

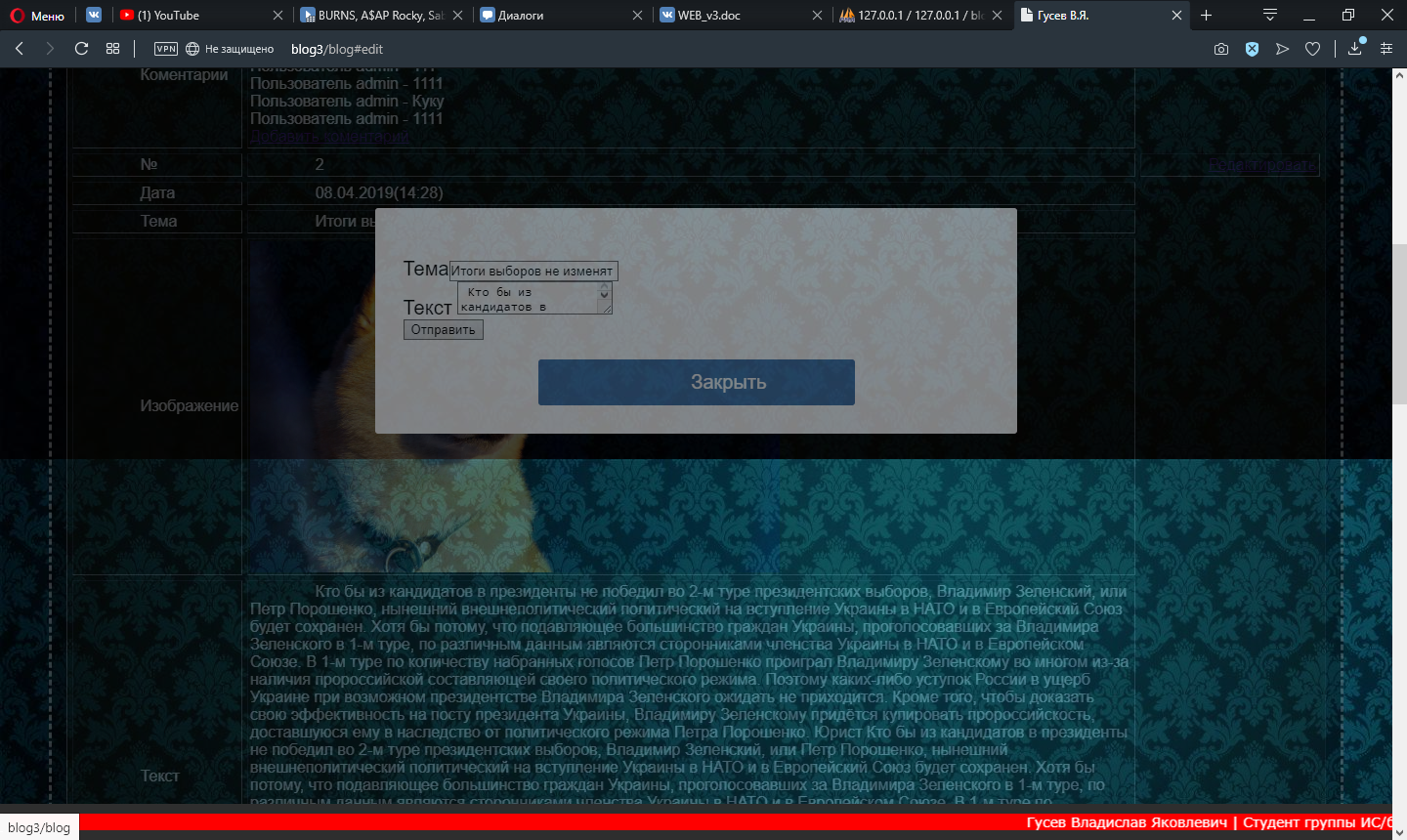


Рисунок 3 –Вид формы редактирования записи блога

Класс coment

<?php

/\*\*

\*

\*/

class coment

{

function addcoment()

{

echo $name\_user="'".$\_SESSION['name']."',";

echo $id\_blog="'".$\_POST['id']."',";

echo $text= "'".$\_POST['text'];

$sql="INSERT INTO users\_comments VALUES (0 , ".$name\_user.$id\_blog.$text."' );";

echo $sql;

BlogModel::get\_q($sql);

echo "Запись успешно добавлена";

echo '<script>window.location = "blog";</script>';

}

}

if(isset($\_POST['actionFunc2'])){

$action\_func = $\_POST['actionFunc2'];

if (function\_exists(coment::$action\_func())){echo "2s";

$action\_func(); //Здесь вызов функции в нашем случае PlusTime;

}

else{

//Если нет вашего метода то какая то логика

}

}

else{

//Если нет скрытого поля то какая то другая логика

}

Класс coment\_model

<?php

include\_once 'app/core/BaseActiveRecord.php';

class CommentsModel extends BaseActiveRecord

{

// список полей таблицы

public $id;

public $name\_user;

public $id\_blog;

public $text;

public static $table = 'users\_comments';

public static function GetParams(){

$st="(id,

id\_user,

id\_blog,

text)";

return $st;

}

}

// делаем PDO доступным для всех AR классов

BaseActiveRecord::$pdo = $pdo;

// получаем запись из БД с id = 1

//$test = TestModel::find(1);

//var\_dump($test->id);

//var\_dump($test->group);

//var\_dump($test->first);

//var\_dump($test->three);

//var\_dump($test);

Класс модели пользователя

<div id="edit" class="modalbackground" >

<div class="modalwindow">

<form method="POST">

<br>

<input hidden id="idedit" type="text" name="id"></input>

Тема<input id="themeedit" type="text" name="theme"></input><br>

Текст <textarea id="textedit" type="text" name="text"></textarea>

<br>

<input type="submit" name="accept">

<?php

require\_once 'app/models/Model\_Blog.php';

if( isset( $\_POST['accept'] ) ){

$q = "UPDATE myblog SET theme='".$\_POST[theme]."', text = '".$\_POST[text]."' WHERE id = $\_POST[id]";

echo $q;

BlogModel::get\_q($q);

echo '<script>window.location = "blog";</script>';

}

?>

</form>

<a href=" ">Закрыть</a>

</div>

</div>

<script type="text/javascript">

/\* Функция, создающая экземпляр XMLHTTP \*/

function getXmlHttp() {

var xmlhttp;

try {

xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");

} catch (e) {

try {

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

} catch (E) {

xmlhttp = false;

}

}

if (!xmlhttp && typeof XMLHttpRequest!='undefined') {

xmlhttp = new XMLHttpRequest();

}

return xmlhttp;

}

function getblog(id) {

var xmlhttp = getXmlHttp(); // Создаём объект XMLHTTP

xmlhttp.open("POST", 'ajaxedit', true); // Открываем асинхронное соединение

xmlhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded'); // Отправляем тип содержимого

xmlhttp.send("id=" + id); // Отправляем POST-запрос

xmlhttp.onreadystatechange = function() { // Ждём ответа от сервера

if (xmlhttp.readyState == 4) { // Ответ пришёл

if(xmlhttp.status == 200) { // Сервер вернул код 200 (что хорошо)

//alert(xmlhttp.responseText);

var str=xmlhttp.responseText.split('/ajaxedit')[1];

var theme= str.split('%')[0];

var text= str.split('%')[1];

document.getElementById("themeedit").value = theme;

document.getElementById("textedit").value = text;

document.getElementById("idedit").value = id;

}

}

};

}

function getblogid(id) {

document.getElementById("id\_coment").value = id;

}

</script>

<div id="addcomment" class="modalbackground" >

<div class="modalwindow">

<?php

require\_once "app/view/coment.php";

?>

<form action="" method="post" target="hiddenframe" enctype="multipart/form-data" >

Текст коментария <textarea id="commentedit" type="text" name="text"></textarea>

<input type="hidden" name="actionFunc2" value="addcoment" />

<input hidden id="id\_coment" type="text" name="id" />

<br>

<input type="submit" name="upload" id="upload" value="Коментировать" />

</form>

<iframe width="1000" height="400" hidden id="hiddenframe" name="hiddenframe" src="">

</iframe>

<?php

?>

</form>

<a href=" ">Закрыть</a>

</div>

</div>

Скрипт регистрации

<script type="text/javascript">

/\* Функция, создающая экземпляр XMLHTTP \*/

function getXmlHttp() {

var xmlhttp;

try {

xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");

} catch (e) {

try {

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

} catch (E) {

xmlhttp = false;

}

}

if (!xmlhttp && typeof XMLHttpRequest!='undefined') {

xmlhttp = new XMLHttpRequest();

}

return xmlhttp;

}

function checkLogin(login) {

var xmlhttp = getXmlHttp(); // Создаём объект XMLHTTP

xmlhttp.open("POST", 'ajax', true); // Открываем асинхронное соединение

xmlhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded'); // Отправляем тип содержимого

xmlhttp.send("login=" + $("#login").val()); // Отправляем POST-запрос

xmlhttp.onreadystatechange = function() { // Ждём ответа от сервера

if (xmlhttp.readyState == 4) { // Ответ пришёл

if(xmlhttp.status == 200) { // Сервер вернул код 200 (что хорошо)

//alert(xmlhttp.responseText);

if (xmlhttp.responseText=="/ajaxoff") document.getElementById("check\_login").innerHTML = "Логин занят!";

else document.getElementById("check\_login").innerHTML = "Логин свободен!";

}

}

};

}

</script>

ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены возможности асинхронного обмена данными с сервером, приобрести практические навыки использования технологии AJAX для организации отправки данных на сервер и получения данных без полной перезагрузки страницы.