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 №2

по дисциплине: «Веб-технологии»

по теме: «Исследование возможностей хранения данных на стороне сервера. Работа с файлами. Работа с реляционными СУБД»

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

(дата)

Руководитель практикума

ст. пр. Забаштанский А.К.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2019

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..

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Разработать базовый класс BaseActiveRecord для работы с базой данных, который реализует паттерн ActiveRecord (данный класс разме-стить в папке /my\_site/app/core).

2. Для всех моделей, которые будут использоваться при выполнении данной лабораторной работы, создать классы, наследующие BaseAc-tiveRecord. Для каждого из классов определить все поля и названия таблиц (данные классы необходимо разместить в папке /my\_site/app/models).

3. Создать новую страницу "Гостевая книга". Страница должна со-держать 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 со-общения. Для хранения сообщений пользователей использовать текстовый файл messages.inc, содержащий разделенные символом «;» данные: Дату сообщения, ФИО, E–mail и Текст отзыва (для получения текущей даты сервера возможно использовать PHP функцию date('d.m.y')).

4. Реализовать страницу "Загрузка сообщений гостевой книги", со-держащую 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 дан-ных 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. 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 добаления, выдаваемые постранично данные.Данные извлекать из таблицы базы данных MySQL. Формат постраничного вывода определяется из Таблицы 3.2 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 за-писей из файла в БД осуществлять с использованием подготавливаемых запросов (см. п. 2.2.4). Валидацию данных осуществлять с использованием класса FormValidation, разработанного при выполнении ЛР №5.

1. ХОД РАБОТЫ

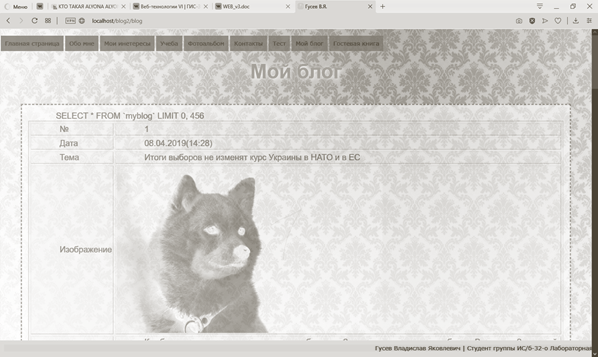


Рисунок 1 – Страница мой блог



Рисунок 2 – Страница Редактор блога

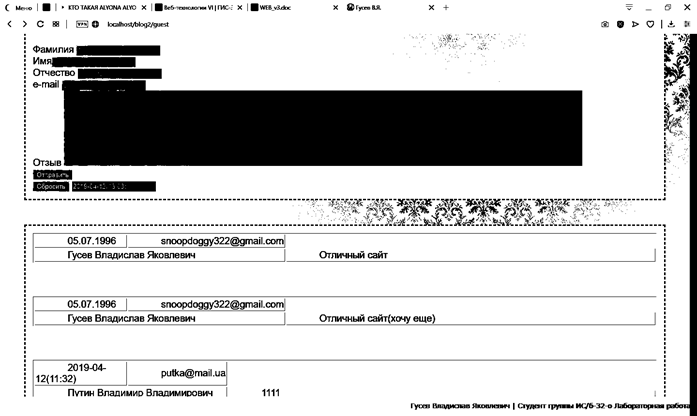


Рисунок 3 – Страница Гостевая книга

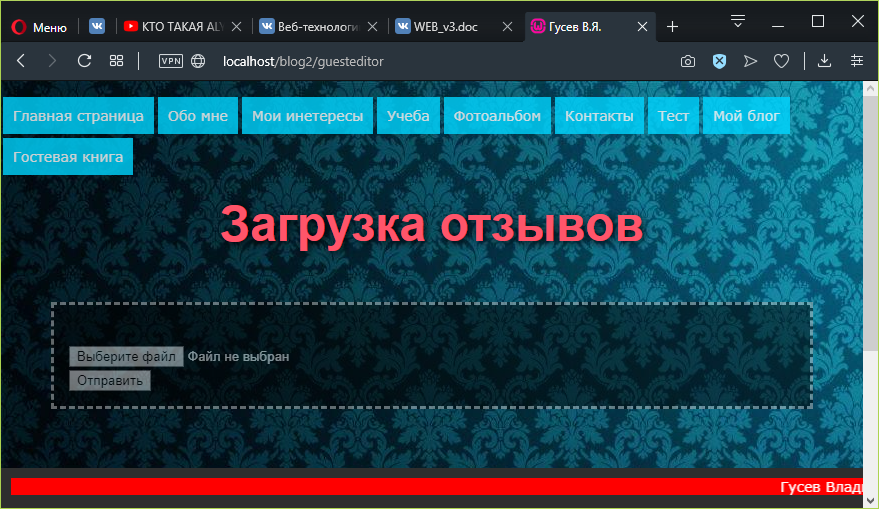


Рисунок 3 – Страница Гостевая книга

Класс BaseActiveRecord -  это подход к доступу к данным в базе данных

<?php

$dsn = 'mysql:dbname=blog;host=127.0.0.1;charset=utf8';

$username = 'root';

$password = '';

try {

$pdo = new PDO($dsn, $username, $password,[PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION]);

} catch (PDOException $e) {

echo 'Подключение не удалось: ' . $e->getMessage();

}

abstract class BaseActiveRecord

{

public static $pdo;

public static function find($id)

{

// подготавливаем SQL запрос

$query = 'SELECT \* FROM '.static::$table.' WHERE id = :id';

$s = self::$pdo->prepare($query);

// подставляем в запрос идентификатор записи,

// которую необходимо извлечь из БД

$s->bindParam(':id', $id);

// получаем запись из БД в виде ассоциативного массива

$s->execute();

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

// если ничего не найдено, то возвращаем NULL

if (!$row) {

return null;

}

// создаем текущий AR объект и возвращаем его в качестве результата

$class = new static();

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

$ar->$key = $value;

}

return $ar;

}

public static function find\_all()

{

// подготавливаем SQL запрос

$query = 'SELECT \* FROM '.static::$table.';';

$s = self::$pdo->query($query);

// подставляем в запрос идентификатор записи,

// которую необходимо извлечь из БД

$data = $s->fetchAll();

return $data;

}

public static function find\_page($n1,$n2)

{

// подготавливаем SQL запрос

$query = 'SELECT \* FROM '.static::$table.' WHERE (id>='.$n1.') AND (id<='.$n2.');';

$s = self::$pdo->query($query);

// подставляем в запрос идентификатор записи,

// которую необходимо извлечь из БД

$data = $s->fetchAll();

return $data;

}

public static function find\_q($q)

{

// подготавливаем SQL запрос

$query = $q;

$s = self::$pdo->query($query);

// подставляем в запрос идентификатор записи,

// которую необходимо извлечь из БД

$data = $s->fetchAll();

return $data;

}

public static function save($param)

{

$str="(";

foreach ($param as $params) {

$str=$str." '".$params."',";

}

$str = substr($str, 0, -1);

$str=$str.")";

if($param[0]!=NULL)

$query = 'SELECT \* FROM '.static::$table." WHERE id=$param[0]";

$query = 'SELECT \* FROM '.static::$table.' WHERE id=666;';

echo $query;

$s = self::$pdo->query($query);

$data = $s->fetchAll();

if(!$data) {

$sql = "INSERT INTO ".static::$table.static::GetParams()." VALUES".$str.";"; }

else {

$query = "DELETE FROM ".static::$table." WHERE id=$param[0];";

$s = self::$pdo->query($query);

$sql="INSERT INTO ".static::$table.static::GetParams()." VALUES".$str.";";

}

echo $sql;

$s = self::$pdo->query($sql);

}

}

Класс blog\_model

<?php

require 'app/core/BaseActiveRecord.php';

class BlogModel extends BaseActiveRecord

{

// список полей таблицы

public $id;

public $date;

public $theme;

public $text;

public $path;

public static $table = 'myblog';

public static function GetParams(){

$st="(id,

date,

theme,

text,

path)";

return $st;

}

function LeftRight($records,$r\_start,$URL,$inpage) {

$str="";

if ($records<=$inpage) return;

if ($r\_start!=0) {

$str.="<a href=".$URL."0>&lt;&lt</a> ";

$str.="<a href=$URL".($r\_start-1).">&lt;</a> ";

}

else $str.="&lt;&lt &lt; ";

if ($r\_start==0) {$sstart=$r\_start-0;$send=$r\_start+10;}

if ($r\_start==1) {$sstart=$r\_start-1;$send=$r\_start+9;}

if ($r\_start==2) {$sstart=$r\_start-2;$send=$r\_start+8;}

if ($r\_start==3) {$sstart=$r\_start-3;$send=$r\_start+7;}

if ($r\_start==4) {$sstart=$r\_start-4;$send=$r\_start+6;}

if ($r\_start>=5) {$sstart=$r\_start-5;$send=$r\_start+5;}

if ($send\*$inpage>$records) $send=$records/$inpage;

if ($sstart<0) $sstart=0;

if ($records%$inpage==0) $add=0; else $add=1;

for ($i=$sstart;$i<$send;$i++) {

if ($i==$r\_start) $str.=" <B>".($i+1)."/".(intval($records/$inpage)+$add)."</B> | ";

else $str.="<a href=$URL".($i)."><U><B>".($i+1)."</B></U></a> | ";

}

if ($r\_start+(1-$add)<intval($records/$inpage)) {

$str.=" <a href=$URL".($r\_start+1).">&gt;</a>";

$str.=" <a href=$URL".(intval($records/$inpage)-(1-$add)).">&gt;&gt;</a>";

}

else $str.=" &gt; &gt;&gt";

return($str);

}

}

BaseActiveRecord::$pdo = $pdo;

Blog\_view.php – Представление страницы блога

<body>

<link rel="stylesheet" href="public/css/style1.css">

<header>

<h1>Мой блог</h1>

</header>

<div display="inline-block"; class="tex">

<?php

$count = 4;// Количество записей на странице

$page = $\_GET["page"];// Узнаём номер страницы

if (!$page) $page=1;

$shift = $count \* ($page -1);

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

echo $s= "SELECT \* FROM `myblog` LIMIT $shift, $count";

$test = BlogModel::find\_all();

echo $c=count($test);

$test = BlogModel::find\_q($s);

if (count($test)) {

// Open the table

echo '<table border="1" bgcolor=#sdkfj cellspacing="5" cellpadding="2" width="100%" height="100% " text-align= "left"

text-indent= "1" >';

foreach ($test as $tests) {

echo "

<tr>

<td>№</td>

<td> $tests[id]</td>

</tr>

<tr text-align='left'>

<td>Дата</td>

<td>$tests[date]</td>

</tr>

<tr>

<td>Тема</td>

<td>$tests[theme]</td>

</tr>

<tr>

<td>Изображение</td>

<td><img width='60%' align='left' src='$tests[path]' alt='$tests[path]'></td>

</tr>

<tr>

<td>Текст</td>

<td>$tests[text]</td>

<tr>

</tr>

";

}

echo "</table>";

print "<center>".BlogModel::LeftRight($c,$page,"blog?page=",$count)."</center>";

}

?>

</table>

</div>

</body>

Guest\_view.php

<style>

.error {

color : red;

}

</style>

<body>

<link rel="stylesheet" href="public/css/style1.css">

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

<header>

<h1>Гостевая книга</h1>

</header>

<div class="tex">

<form method="POST" enctype="multipart/form-data" name="form\_editor" style="font-size: 20px"><br>

Фамилия <input placeholder="Фамилия" maxlength="100" name="name" type= text class="formDesign" > </input> <br>

Имя<input placeholder=" Имя" maxlength="100" name="name1" type= text class="formDesign" > </input> <br>

Отчество <input placeholder="Отчество" maxlength="100" name="name2" type= text class="formDesign" > </input> <br>

<?php

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

if($model && $model->getErrorByField('name')) {

echo "<div class='error'>" . $model->getErrorByField('name')."</div>";

}

else {

$path = 'public/files/';

$location = $path . "messages.inс";

$handle = fopen($location, "a");

$DateTime = date('Y-m-d(H:i)').",";

$name=$\_POST['name'].",";

$name1=$\_POST['name1'].",";

$name2=$\_POST['name2'].",";

$email=$\_POST['email'].",";

$txt=$\_POST['text']."\n";

$writetext = $DateTime.$name.$name1.$name2.$email.$txt;

if (fwrite($handle, $writetext)) {

header('Location: main');

unset($\_POST);

}

else {

echo "Произошла ошибка при записи данных";

fclose($handle);

}

}

}

?>

e-mail <input placeholder="e-mail" maxlength="100" name="email" type= email class="formDesign" > </input> <br>

Отзыв <textarea type="text" placeholder="Отзыв" rows="10" cols="150" maxlength="1000" name="text" type= text class="formDesign" ></textarea>

<br>

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

<br>

<input type="reset" name="reset">

<input type="f" name="date" value="<?php echo date('Y-m-d(H:i)'); ?> "disabled>

</form>

</div>

<div display="inline-block"; class="tex">

<?php

$path = 'public/files/';

$location = $path . "messages.inс";

$row=1;

if (($handle = fopen($location, "r")) !== FALSE) {

$data = fgetcsv($handle, 1000, ",");

while (($data = fgetcsv($handle, 1000, ",")) !== FALSE) {

$num = count($data);

// echo "<p> $num полей в строке $row: <br /></p>\n";

$row++;

echo '<table border="1" width=100%;>

<tr>

<td style="width: 15%;">'.$data[0].'</td>

<td style="width: 15%;">'.$data[4].'</td>

</tr>

<tr>

<td colspan="2" style="width: 30%";>'.$data[1]." ".$data[2]." ".$data[3].'</td>

<td >'.$data[5].'</td>

</tr>

</table>

<br><br><br>

';

}

fclose($handle);

}

?>

</table>

</div>

</body>

ВЫВОДЫ

В ходе выполнения лабораторной работы были изучены основы синтаксиса PHP и принципы функционирования MVC приложения на стороне сервера. Приобретены практические навыки использования языка PHP для генерации HTML-кода и обработки HTML-форм в MVC приложении.