# ЛАБОРАТОРНЯ РАБОТА №9 «Исследование архитектуры MVC приложения и возможностей обработки данных HTML-форм на стороне сервера с использованием языка PHP»

## Цель работы

Исследовать возможности хранения данных на стороне сервера, изучить 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 постраничного вывода данных.

## Порядок выполнения работы

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

2. Для всех моделей, которые будут использоваться при выполнении данной лабораторной работы, создать классы, наследующие BaseActiveRecord. 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 содержать 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 сообщения. 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", содержащую 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 данных 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. Форма добавления должна содержать поля ввода: • Тема сообщения – поле ввода однострочного текста (заполнение обязательно); • Изображение – поле ввода файла (заполнение не обязательно); • Текст сообщения – поле ввода многострочного текста. Данные хранить в разработанной таблице базы данных MySQL. Валидацию данных осуществлять с использованием класса FormValidation, разработанного при выполнении ЛР №8. 7. Разработать страницу «Мой Блог», содержащую упорядоченные в порядке убывания даты добавления, выдаваемые постранично данные: • Дата и время сообщения;

• Тема сообщения; • Изображение; • Текст сообщения. Данные извлекать из таблицы базы данных MySQL. Формат постраничного вывода определяется из таблицы 2.2 в соответствии с вариантом задания. Вариант задания определяется как (ПЦЗК mod 6)+1 – (остаток от деления последней цифры зачетной книжки на 6 плюс 1).

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 записей 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 №8.

## Ход работы

### Разработанный базовый класс BaseActiveRecord для работы с базой данных, представлен на рисунке 2.1. Полная реализации представлена в приложении А.

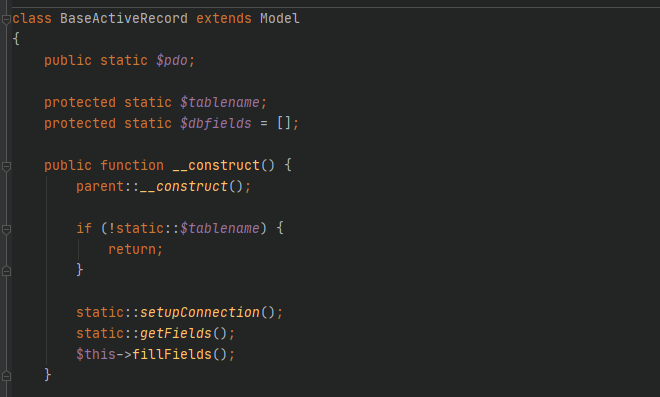


Рисунок 2.1 – Реализация класса BaseActiveRecord

### На рисунке 2.2 представлена разработанная страница гостевой книги. На рисунке 2.3 содержимое файла messages.inc



Рисунок 2.2 – Страница гостевой книги

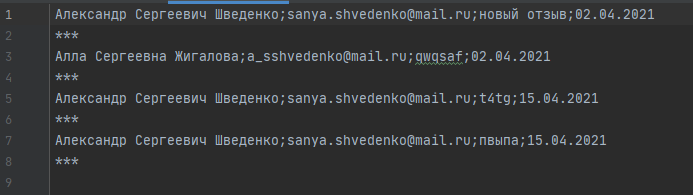


Рисунок 2.3 – Страница гостевой книги

### На рисунке 2.4 представлена разработанная страница загрузки сообщений гостевой книги.

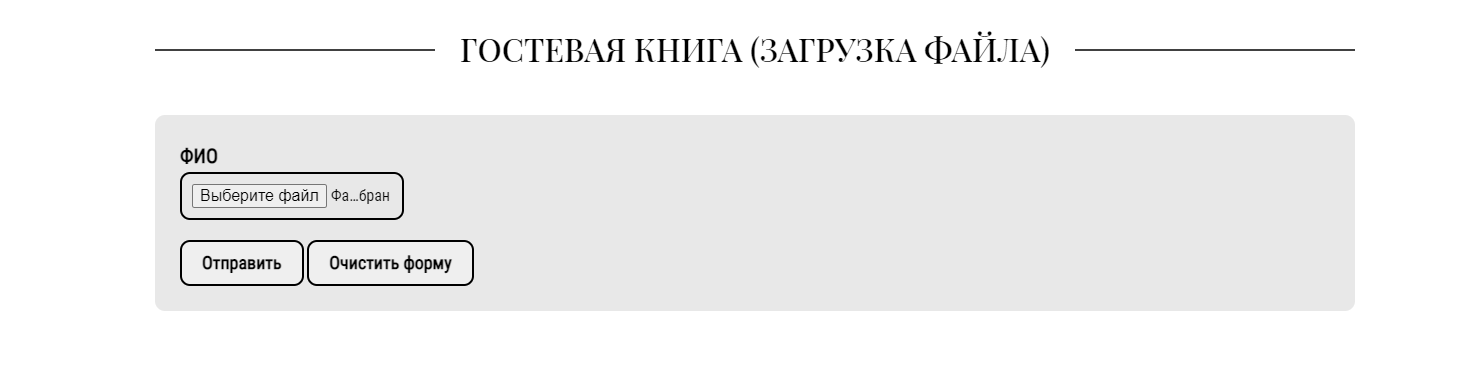


Рисунок 2.4 – Страница загрузки сообщений гостевой книги

### Реализация метода сохранения данных из страницы Тест по дисциплине представлена на рисунке 2.5

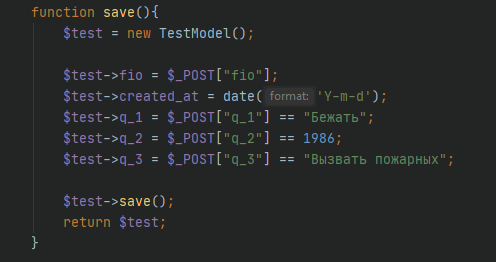


Рисунок 2.5 – Реализация метода save

### На рисунке 2.6 представлена разработанная страница редактирования блога

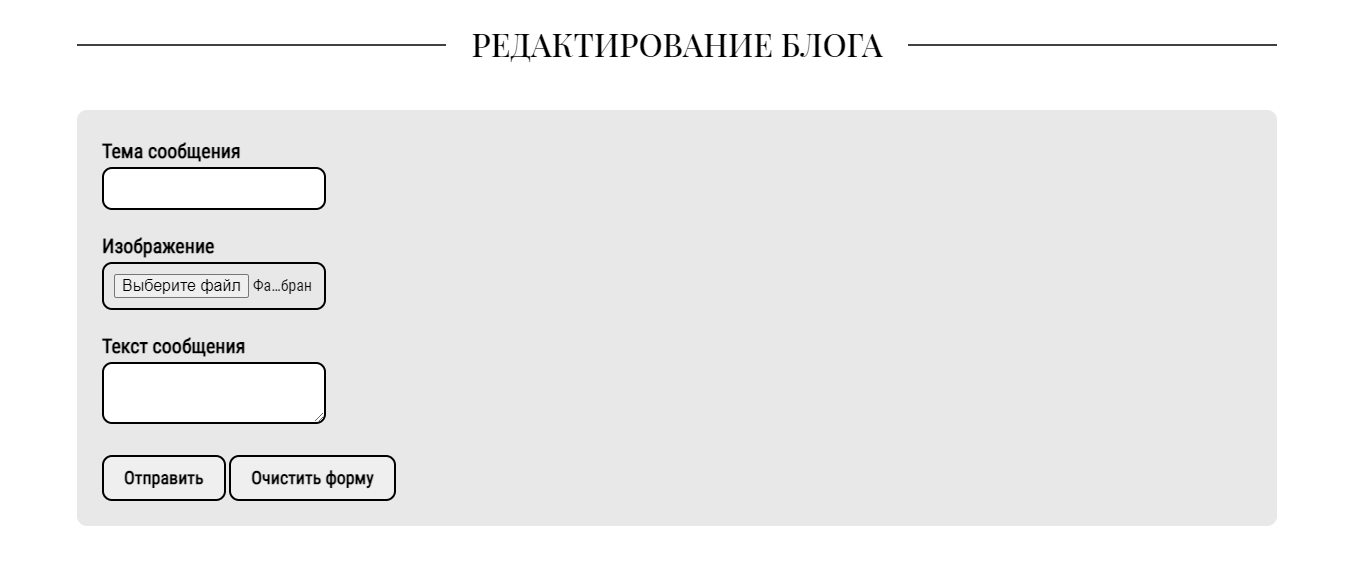


Рисунок 2.6 – Страница редактирования блога

### На рисунке 2.7 представлена разработанная страница блога с постраничной навигацией.

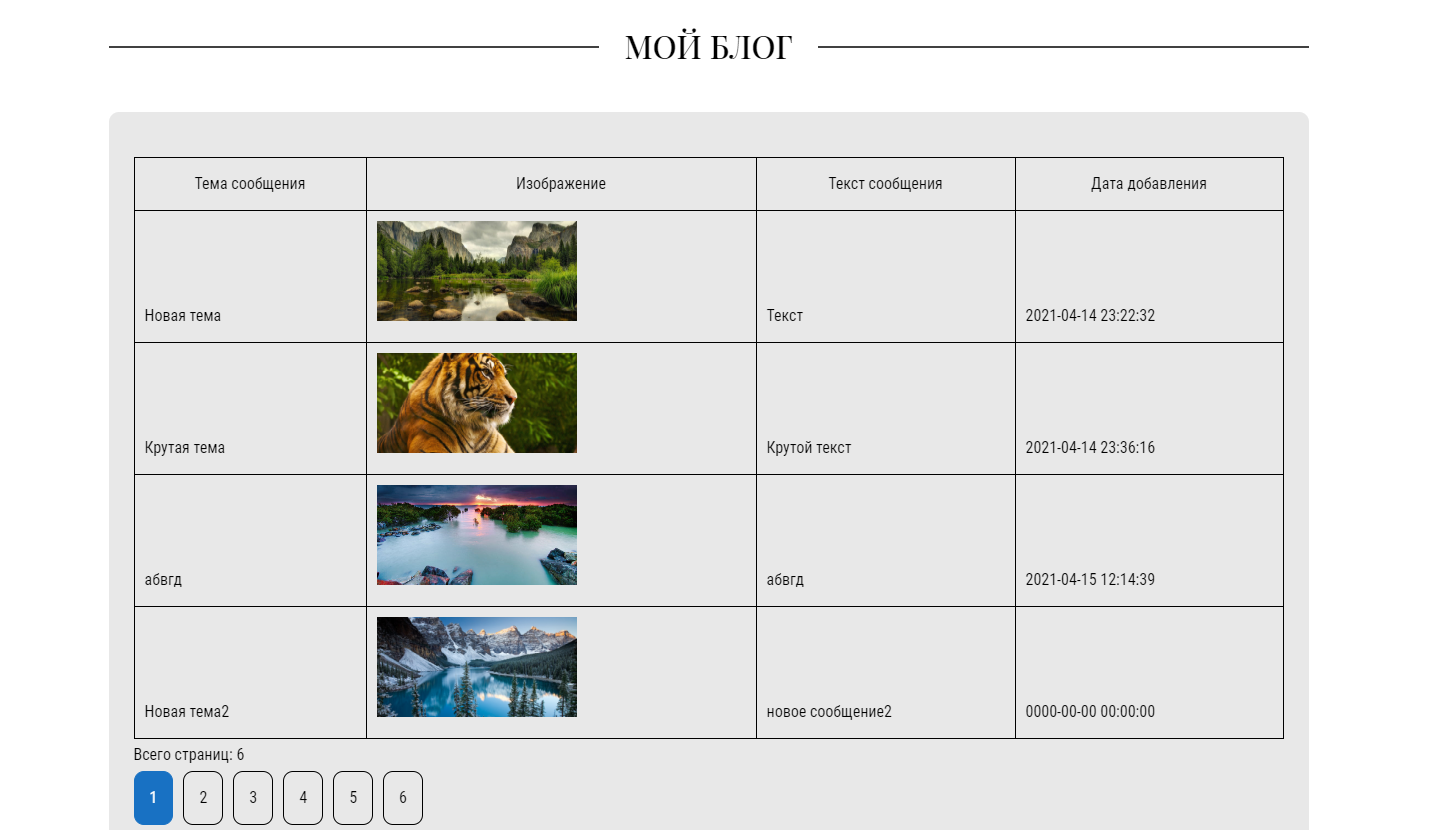


Рисунок 2.7 – Страница блога с постраничной навигацией

### На рисунке 2.8 представлена разработанная страница загрузки сообщений для блога. На рисунке 2.9 представлено содержимое файла csv

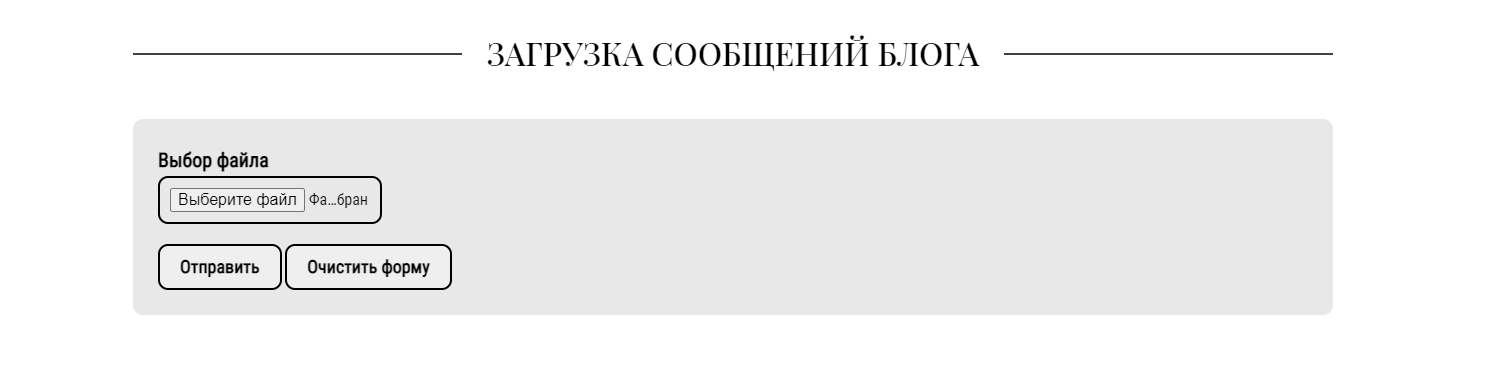


Рисунок 2.8 – Страница загрузки сообщений

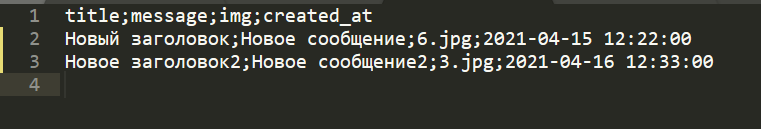


Рисунок 2.9 – Содержимое файла csv

# Вывод

В ходе выполнения данной лабораторной работы был изучен шаблон проектирования ActiveRecord. На его основе был составлен класс BaseActiveRecord. Он позволяет работать с базой данных через ООП. Также была изучена работа с файлами на стороне сервера с помощью языка программирования PHP и его суперглобальной переменной $\_FILES. Кроме этого была реализована постраничная навигация для вывода записей со страницы «Мой блог», а также загрузка сообщений через файл формата CSV.

# Приложение А

Файл core/BaseActiveRecord.php

<?php  
  
  
namespace app\core;  
  
use PDO;  
use PDOException;  
  
class BaseActiveRecord extends Model  
{  
 public static *$pdo*;  
  
 protected static *$tablename*;  
 protected static *$dbfields* = [];  
  
 public function \_\_construct() {  
 parent::*\_\_construct*();  
  
 if (!static::*$tablename*) {  
 return;  
 }  
  
 static::*setupConnection*();  
 static::*getFields*();  
 $this->fillFields();  
 }  
  
 public static function getFields() {  
 $stmt = static::*$pdo*->query("SHOW FIELDS FROM " . static::*$tablename*);  
 while ($row = $stmt->fetch()) {  
 static::*$dbfields*[$row['Field']] = $row['Type'];  
 }  
 }  
  
 public function fillFields(){  
 foreach (static::*$dbfields* as $field => $type){  
 $this->$field;  
 }  
 }  
  
 public static function setupConnection() {  
 if (!isset(static::*$pdo*)) {  
 try {  
 static::*$pdo* = new PDO("mysql:dbname=Web-LR; host=localhost; charset=utf8", "root", "");  
 } catch (PDOException $ex) {  
 die("Ошибка подключения к БД: $ex");  
 }  
 }  
 }  
  
 public static function find($id) {  
 $sql = "SELECT \* FROM " . static::*$tablename* . " WHERE id=$id";  
 $stmt = static::*$pdo*->query($sql);  
 $row = $stmt->fetch(PDO::*FETCH\_ASSOC*);  
  
 if (!$row) return null;  
  
 $ar\_obj = new static();  
  
 foreach ($row as $key => $value) {  
 $ar\_obj->$key = $value;  
 }  
  
 return $ar\_obj;  
 }  
  
 public static function all($orderBy = "id", $orderDirection = "ASC"){  
 $sql = "SELECT \* FROM " . static::*$tablename* . " ORDER BY $orderBy $orderDirection";  
 static::*get*($sql);  
 }  
  
 private static function get($sql){  
 $stmt = static::*$pdo*->query($sql);  
 $tableData = $stmt->fetchAll(PDO::*FETCH\_ASSOC*);  
  
 if (!$tableData) return null;  
  
 $records = [];  
  
 foreach ($tableData as $row) {  
 $ar\_obj = new static();  
  
 foreach ($row as $key => $value) {  
 $ar\_obj->$key = $value;  
 }  
  
 $records[] = $ar\_obj;  
 }  
  
 return $records;  
 }  
  
 public static function paginate($numPage){  
 if (isset($\_GET['page'])){  
 $page = $\_GET['page'];  
 } else $page = 1;  
 $recordStart = ($page \* $numPage) - $numPage;  
  
 $sql = "SELECT \* FROM " . static::$tablename . " LIMIT $numPage OFFSET $recordStart";  
 return static::get($sql);  
 }  
  
 public static function getNumRow(){  
 $sql = "SELECT COUNT(\*) as count FROM " . static::$tablename;  
 return static::$pdo->query($sql)->fetchColumn();  
 }  
  
 public function save() {  
 foreach (static::$dbfields as $field => $type){  
 $holder = $this->$field;  
 static::$dbfields[$field] = $holder;  
 }  
  
 $dbfields = "`" . implode('`, `', array\_keys(static::$dbfields)) . "`";  
 $prepareStr = ":" . implode(', :', array\_keys(static::$dbfields));  
 unset(static::$dbfields['id']);  
 static::$dbfields['id'] = NULL;  
  
 $sql = "INSERT INTO " . static::$tablename . " ($dbfields) VALUES ($prepareStr)";  
 static::$pdo->prepare($sql)->execute(static::$dbfields);  
 }  
  
 public function delete() {  
 $sql = "DELETE FROM " . static::$tablename . " WHERE ID=" . $this->id;  
 $stmt = static::$pdo->query($sql);  
 if ($stmt) {  
 return $stmt->fetch(PDO::FETCH\_ASSOC);  
 } else {  
 print\_r(static::$pdo->errorInfo());  
 }  
 }  
}