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 №1

по дисциплине: «Веб-технологии»

по теме: «Исследование архитектуры MVC приложения и возможностей

обработки данных HTML-форм на стороне сервера с использованием языка

PHP.»

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

(дата)

Руководитель практикума

ст. пр. Забаштанский А.К.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2019

1. ЦЕЛЬ РАБОТЫ

Изучить основы синтаксиса PHP и принципы функционирования MVC приложения на стороне сервера. Приобрести практические навыки использования языка PHP для генерации HTML-кода и обработки HTML-форм в MVC приложении.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ
2. ХОД РАБОТЫ
3. В соответствии с рекомендациями, представленными в п. 1.2.8 реструктурировать код персонального сайта на основе шаблона MVC.
4. Для страницы «Фотоальбом» реализовать хранение данных фо-то(подписей и имен файлов) в соответствующих константах модели Photo. Вывод таблицы, содержащей фото, реализовать в представлении с использованием операторов циклов.
5. Для страницы «Мои интересы» реализовать хранение данных (название категорий, интересов, описание интересов) в соответствующих константах модели Interest. Вывод меню ссылок и интересов реализовать в представлении с использованием операторов циклов.
6. Реализовать класс FormValidation, выполняющий валидацию данных форм, передаваемых на сторону сервера.
7. С использованием разработанного класса реализовать валидацию форм «Контакт» и «Тест по дисциплине “…”».
8. Реализовать дочерний класс CustomFormValidation от класса FormValidation, дополнив его возможностью выполнения специализированной проверки формы «Тест по дисциплине» на стороне сервера.

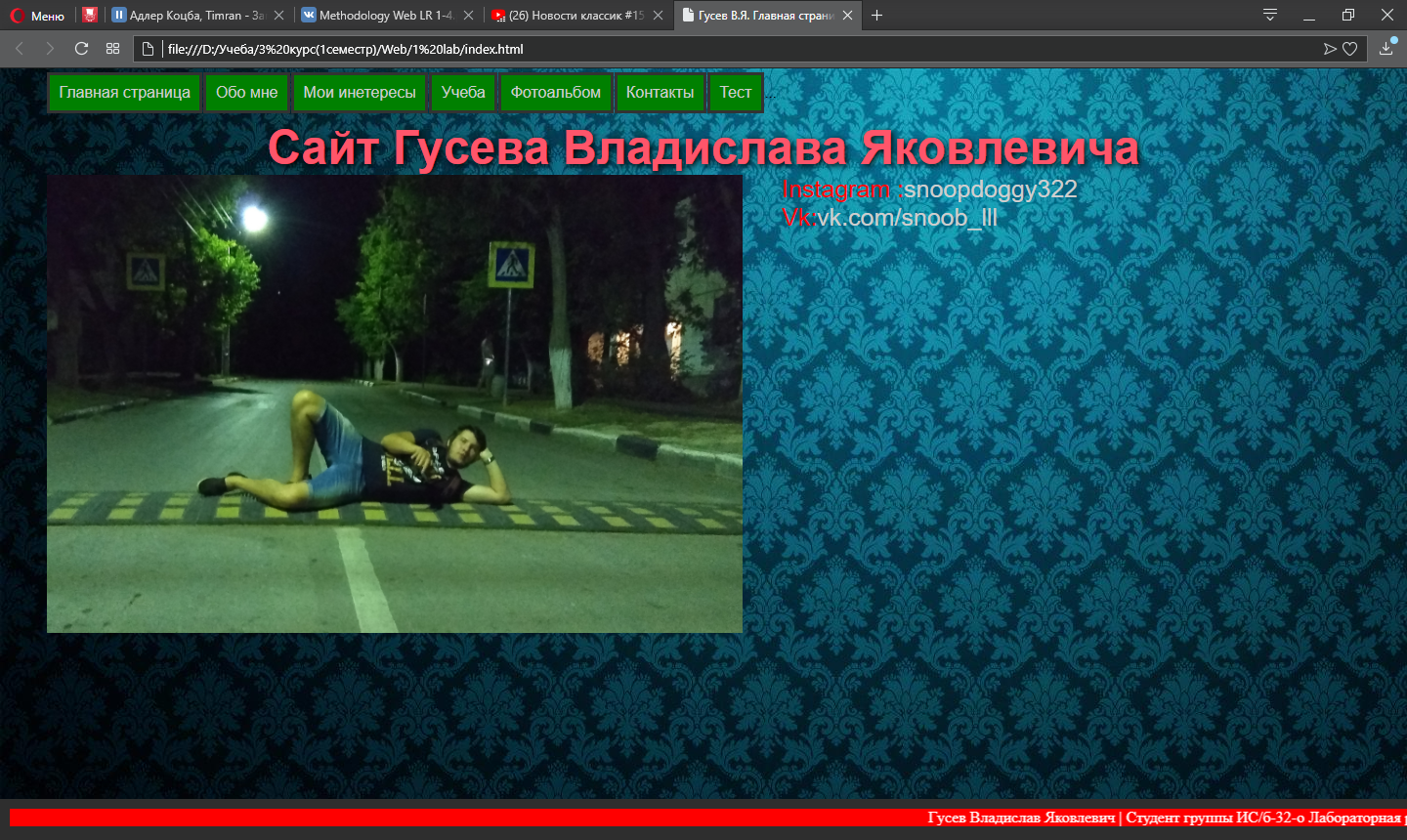


Рисунок 1 – Главная страница

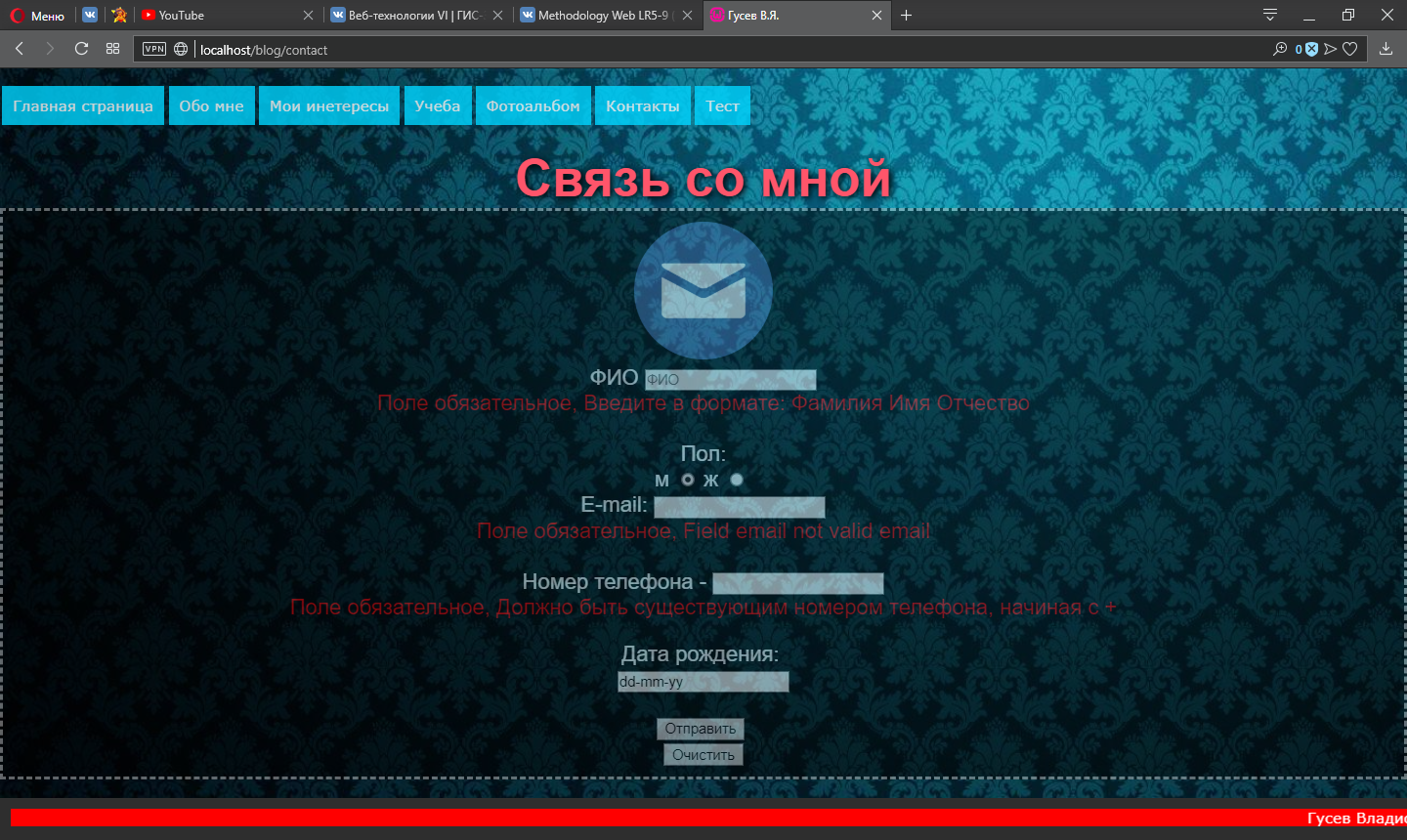


Рисунок 2 – Валидация формы «Контакт»

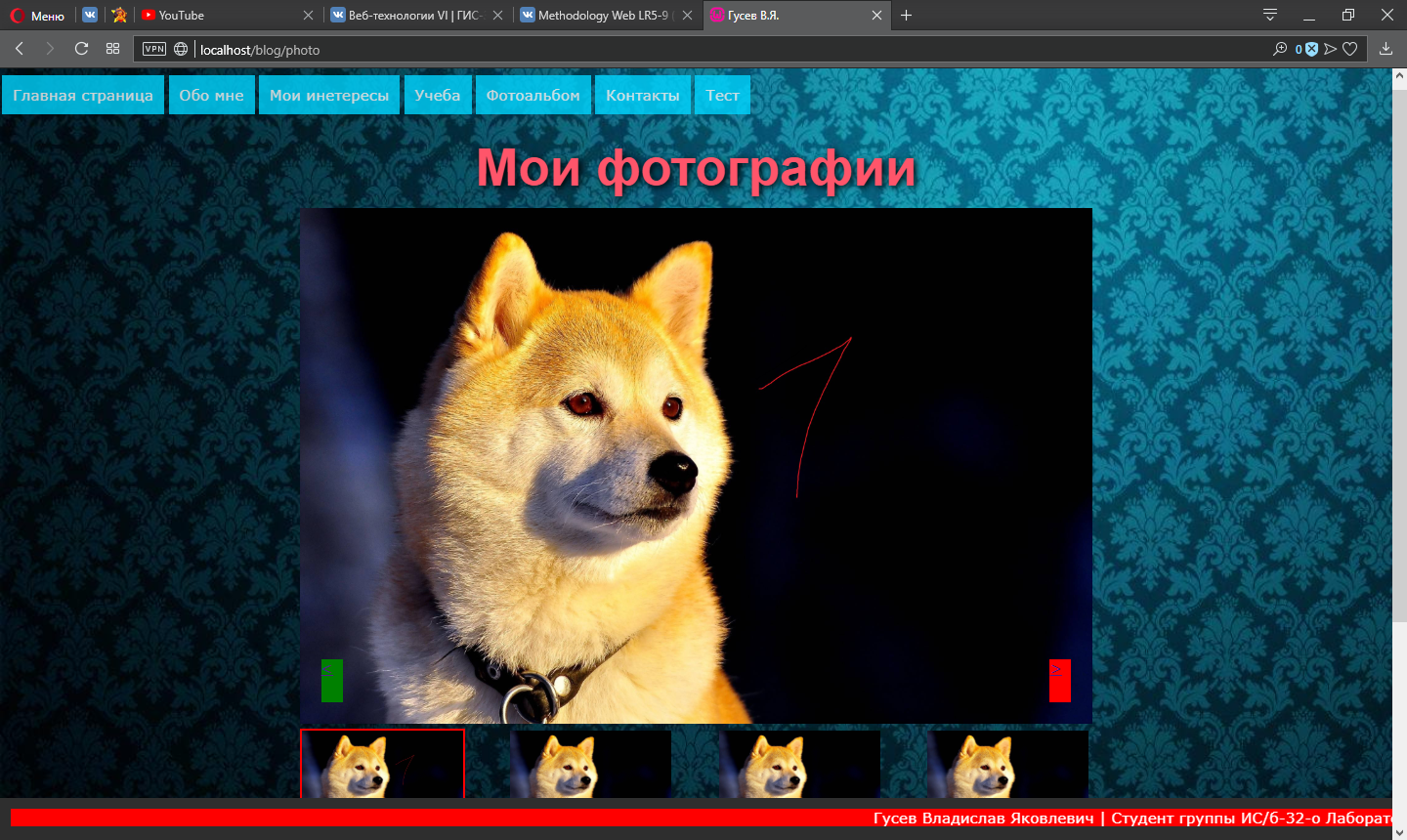


Рисунок 3 – Работа галереи

Код файла index.php – Точка входа в программу.

<?php

ini\_set('display\_errors', 0);

require\_once 'app/bootstrap.php';

Код файла .htacces - обработка запроса в при открытии страницы.

RewriteEngine On

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule .\* index.php [L]

Файл Controller.php - базовый класс контроллера приложения.

<?php

class Controller {

public $model;

public $view;

function \_\_construct()

{

$this->view = new View();

}

function action\_index()

{

}

}

Файл route.php - класс роутера приложения.

<?php

spl\_autoload\_register(function ($name) {

echo "Хочу загрузить $name.\n";

throw new Exception("Невозможно загрузить $name.");

});

class Route

{

static function start()

{

// контроллер и действие по умолчанию

$controller\_name = 'Main';

$action\_name = 'index';

$routes = explode('/', $\_SERVER['REQUEST\_URI']);

// получаем имя контроллера

if ( !empty($routes[2]) )

{

$controller\_name = $routes[2];

}

// получаем имя экшена

if ( !empty($routes[3]) )

{

$action\_name = $routes[3];

}

// добавляем префиксы

$model\_name = 'Model\_'.$controller\_name;

$controller\_name = 'Controller\_'.$controller\_name;

$action\_name = 'action\_'.$action\_name;

$model\_file = strtolower($model\_name).'.php';

$model\_path = "application/models/".$model\_file;

if(file\_exists($model\_path))

{

include "application/models/".$model\_file;

}

// подцепляем файл с классом контроллера

$controller\_file = strtolower($controller\_name).'.php';

$controller\_path = "application/controllers/".$controller\_file;

if(file\_exists($controller\_path))

{

include "application/controllers/".$controller\_file;

}

else

{

//Route::ErrorPage404();

}

}

// создаем контроллер

include('app/controllers/'.$controller\_name.'.php');

$controller = new $controller\_name;

$action = $action\_name;

if(method\_exists($controller, $action))

{

// вызываем действие контроллера

$controller->$action();

}

else

{

}

}

function ErrorPage404()

{

$host = 'http://'.$\_SERVER['HTTP\_HOST'].'/';

header('HTTP/1.1 404 Not Found');

header("Status: 404 Not Found");

header('Location:'.$host.'404');

}

}

Файл view.php - базовый класс представления приложения

<?php

class View

{

function generate($content\_view, $template\_view,$model = NULL, $data = null)

{

include 'app/view/'.$template\_view;

}

}

?>

Файл controller\_contact.php – контроллер страницы контакт.

<?php

require 'app/models/Validation.php';

class Controller\_contact extends Controller

{

protected $\_rules = [

'name' =>'required,words:2-3',

'email' =>'required,email',

'tel' =>'required,phone',

];

protected $\_messages = [

'name' => [

'required'=>'Поле обязательное',

'words' =>'Введите в формате: Фамилия Имя Отчество',

],

'tel' => [

'required'=>'Поле обязательное',

'phone' =>'Должно быть существующим номером телефона, начиная с +',

],

'email' => [

'required'=>'Поле обязательное',

'phone' =>'Должно быть существующим email',

],

];

function action\_index()

{

$validator = new Validation($\_POST, $this->\_rules, $this->\_messages);

$validator->validate();

$this->view->generate('contact\_view.php', 'template\_view.php', $validator);

}

public function submit()

{

$validator = new Validation($\_POST, $this->\_rules, $this->\_messages);

$validator->validate();

if($validator->isValid()) {

$this->view->render('contact\_send.php', 'template\_view.php');

} else {

$this->view->render('contact', 'contact', $validator);

}

}

}

Файл controller\_test.php – класс контроллера

<?php

require 'app/models/Validation.php';

class Controller\_test extends Controller

{

protected $\_rules = [

'name' =>'required,words:2-3',

'group' =>'required',

'random\_value' =>'required',

'mark' =>'required,int',

'event\_'=>'count:1-3',

];

protected $\_messages = [

'name' => [

'required'=>'Поле обязательное',

'words' =>'Введите в формате: Фамилия Имя Отчество',

],

'group' =>'Поле обязательное',

'points' =>'Поле обязательное',

'event\_'=>'Должно быть выбрано 1-3 пунктов',

'mark' => [

'required'=>'Поле обязательное',

'int' =>'Должно быть числом',

],

];

function action\_index()

{

$validator = new Validation($\_POST, $this->\_rules, $this->\_messages);

$validator->validate();

$this->view->generate('test\_view.php', 'template\_view.php',$validator);

}

public function submit()

{

$validator = new Validation($\_POST, $this->\_rules, $this->\_messages);

$validator->validate();

if($validator->isValid()) {

$this->view->render('test\_complete', 'Test complete' );

} else {

$this->view->render('test', 'Test', $validator);

}

}

}

Файл form\_validation.php – класс валидатора формы.

<?php

require 'app/models/validators/form\_validation.php';

class Model

{

public $validator;

function \_\_construct()

{

$this->validator = new FormValidation();

}

function validate($post\_data)

{

$this->validator->validate($post\_data);}}

Файл Validation.php – класс реализующий все методы валидации.

<?php

class Validation

{

protected $\_rules = [];

protected $\_errors = [];

protected $\_form = [];

protected $\_messages = [];

protected $\_aliases = [

'required' => 'notEmpty',

'integer' => 'int',

'gt' => 'greater',

'lt' => 'less',

'tel' => 'phone',

];

protected $\_result = '';

public function \_\_construct(array $form, array $rules = [], array $messages = [])

{

$this->\_form = $form;

$this->\_rules = $this->parseRules($rules);

$this->\_messages = $messages;

}

protected function parseRules($rules)

{

$parsed = [];

foreach ($rules as $field => $field\_rules) {

if (!is\_array($field\_rules)) {

$field\_rules = explode(',', $field\_rules);

}

$parsed[$field] = $field\_rules;

}

return $parsed;

}

public static function run(array $rules, array $messages = [])

{

$validator = new static($\_REQUEST, $rules, $messages);

$validator->validate();

return $validator;

}

public function validate()

{

foreach ($this->\_rules as $field => $field\_rules) {

foreach ($field\_rules as $rule) {

if (!$this->validateRule($field, $rule)) {

if (empty($this->\_errors[$field])) {

$this->\_errors[$field] = [];

}

$this->\_errors[$field][] = $this->errorMessage($field, $rule);

}

}

}

if (empty($this->\_errors)) {

$this->verify();

}

return $this;

}

protected function validateRule($field, $rule)

{

$tmp = explode(':', $rule);

$value = null;

if (count($tmp) > 1) {

$rule = $tmp[0];

$value = array\_slice($tmp, 1, count($tmp) - 1);

}

$method = 'is' . ucfirst($rule);

if (!method\_exists($this, $method) && !empty($this->\_aliases[$rule])) {

$method = 'is' . ucfirst($this->\_aliases[$rule]);

}

if (method\_exists($this, $method)) {

return call\_user\_func([$this, $method], $field, $value);

}

return true;

}

public function isValid()

{

return count($this->\_errors) === 0;

}

public function getErrorByField($name)

{

if(isset($this->\_errors[$name]))

return implode(', ', $this->\_errors[$name]);

return null;}

protected function errorMessage($field, $rule)

{

$tmp = explode(':', $rule);

$rule = $tmp[0];

$value = count($tmp) > 1 ? $tmp[1] : null;

$message = 'Field ' . $field . ' not valid ' . $rule;

if (!empty($this->\_messages[$field])) {

if (!is\_array($this->\_messages[$field])) {

$message = $this->\_messages[$field];

} elseif (!empty($this->\_messages[$field][$rule])) {

if (!is\_array($this->\_messages[$field][$rule])) {

$message = $this->\_messages[$field][$rule];

} elseif (!empty($this->\_messages[$field][$rule][$value])) {

$message = $this->\_messages[$field][$rule][$value];

}

}

}

return $message;

}

public function verify()

{

}

public function errors()

{

return $this->\_errors;

}

public function result()

{

return $this->\_result;

}

public function isNotEmpty($field)

{

return !empty($this->\_form[$field]);

}

public function isChecked($field)

{

return !empty($this->\_form[$field]) && $this->\_form[$field] == true;

}

public function isEmail($field)

{

return preg\_match('/.+@.+\..+/', $this->\_form[$field]);

}

public function isLess($field, $value)

{

return $this->isInt($field) && $this->\_form[$field] < $value;

}

public function isInt($field)

{

return !empty($this->\_form[$field]) && (is\_int($this->\_form[$field]) || is\_numeric($this->\_form[$field]));

}

public function isGreater($field, $value)

{

return $this->isInt($field) && $this->\_form[$field] < $value;

}

public function isWords($field, $value)

{

if (!empty($this->\_form[$field])) {

$count = [1, 1];

if (!empty($value)) {

$tmp = array\_map(function ($v) {

return (int)$v;

}, explode('-', $value[0]));

if (count($tmp) === 2) {

$count = $tmp[0] === $tmp[1] ?

$tmp[0] :

($tmp[0] <= $tmp[1] ?

[$tmp[0], $tmp[1]] :

[$tmp[1], $tmp[0]]);

} elseif (is\_int($tmp[0])) {

$count = [(int)$tmp[0], (int)$tmp[0]];

}

}

if ($count[0] < 1) {

return false;

}

$regex = sprintf('/([a-z0-9а-яё]+\s+){%d,%d}[a-z0-9а-яё]+/i', $count[0] - 1, $count[1] - 1);

return preg\_match($regex, $this->\_form[$field]);

}

return false;

}

public function isDate($field, $value)

{

return date\_create\_from\_format(empty($value) ? 'd.m.Y' : $value[0], $this->\_form[$field]) !== false;

}

public function isPhone($field)

{

return preg\_match('/^\+?(?:380(\d){9}|7\s\*\(?\d{3}\)?\s\*(?:-?\d){7})$/i', $this->\_form[$field]);

}

public function setRule($field, $rule)

{

if (empty($this->\_rules[$field])) {

$this->\_rules[$field] = [];

}

$this->\_rules[$field][] = $rule;

return $this;

}

}

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.