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 №5

по дисциплине «Веб-технологии»

на тему «Программирование на стороне сервера с использованием языка PHP. Исследование возможностей обработки данных HTML-форм.»

Отметка о зачете _____ _____
(дата)

Руководитель практикума

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

Севастополь 2017

1. ЦЕЛЬ РАБОТЫ

Изучить основы синтаксиса PHP, приобрести практические навыки использования управляющих конструкций, операторов и функций в PHP для генерации HTML-кода и обработки HTML-форм. Приобрести практические навыки обработки строк в PHP-скриптах, а также работы с файлами для хранения данных на стороне сервера. Приобрести практические навыки создания PHP-скриптов с использованием парадигм ООП.

2. ПОСТАНОВКА ЗАДАЧИ

2.1. Реструктурировать код приложения по принципу MVC паттерна (разделить бизнес логику и отображение). Для этого необходимо составить такую структуру файлов и каталогов:

Каталог/файл	**Описание**
/my_site/public/assets/js/	Каталог содержит все JavaScript файлы приложения
/my_site/public/assets/css/	Каталог содержит все CSS файлы приложения
/my_site/public/assets/img/	Каталог содержит все изображения приложения
/my_site/public/index.php	Корневая и единственная точка входа в приложение.
/my_site/app/controllers/	Содержит PHP классы, которые обрабатывают HTTP запросы
/my_site/app/views/	Содержит PHP файлы с PHP и HTML кодом. В эту папку переносятся все HTML шаблоны из предыдущих лабораторных работ. Также для шаблонов из предыдущих лабораторных меняется расширение с HTML на PHP.
/my_site/app/core	Содержат классы необходимые для работы приложения

2.2. Необходимо разработать класс Dispatcher.php и набор правил перенаправления (rewrite rules) для сервера Apache (или Nginx), которые позволят получать доступ к страницам сайта в «человекоподобном формате». Например:

my_site/user/photos/ - страница «Фотоальбом»

my_site/user/hobby/ - страница «Мои интересы»

и т.д.

Для реализации правил перенаправления на стороне сервера необходимо воспользоваться онлайн документацией. Таким образом, цикл работы PHP приложения будет иметь вид:

- Запрос my_site/user/photos/
- Сервер Apache/Nginx вызывает index.php?route=user/photos
- index.php вызывает класс Dispatcher.php
- Класс Dispatcher.php вызывает метод класса UsersController::photos
- Метод UsersController::photos формирует переменные и вызывает файл app/views/users_photos.php для отображения списка фотографий

Как видно из указанного алгоритма каждое звено выполняет только одну определенную ему роль. Диспетчер – маршрутизация, контроллер – передача данных для отображения, view – отображение HTML.

2.3. Модифицировать страницу «Фотоальбом», реализовав PHP-функцию вывода таблицы, содержащей фото (с использованием операторов циклов). Значения имен файлов фото и подписей к фото предварительно разместить в глобальных массивах *fotos* и *alts*.

2.4. Модифицировать страницу «Мои интересы», реализовав вывод списков с использованием PHP-функции с переменным числом аргументов.

2.5. Реализовать класс FormValidation, реализующий валидацию данных форм, передаваемых на сторону сервера. Рекомендуемая структура класса:

Rules – поле(массив), содержащее набор правил для проверки валидности данных;

Errors – поле(массив), содержащее тексты ошибок возникших при проверке валидности данных;

isNotEmpty(data) – метод проверки является ли значение data не пустым – возвращает сообщение об ошибке, если таковая имеется;

isInteger(data) – метод проверки является ли значение data строковым представлением целого числа – возвращает сообщение об ошибке, если таковая имеется;

isLess(data, value) – метод проверки является ли значение data строковым представлением целого числа и не меньшим, чем value – возвращает сообщение об ошибке, если таковая имеется;

isGreater(data, value) – метод проверки является ли значение data строковым представлением целого числа и не большим, чем value – возвращает сообщение об ошибке, если таковая имеется;

isEmail(data) – метод проверки является ли значение data строковым представлением email – возвращает сообщение об ошибке, если таковая имеется;

SetRule(field_name, validator_name) – метод, добавляющий в массив Rules проверку для поля field_name типа validator_name;

Validate(post_array) – метод выполняющий проверку элементов в массиве post_array, в соответствии с правилами Rules и сохраняющий сообщения об ощибках в поле Errors;

ShowErrors() – метод выводящий все сообщения об ошибках из поля Errors в формате HTML.

2.6. С использованием разработанного класса реализовать валидацию форм «Контакт» и «Тест по дисциплине “…”».

2.7. Реализовать дочерний класс CustomFormValidation от класса FormValidation, дополнив его возможностью выполнения специализированной проверки формы «Тест по дисциплине» на стороне сервера.

2.8. Реализовать дочерний класс ResultsVerification от класса CustomFormValidation, дополнив его возможностью проверки правильности ответов, введенных пользователем на странице "Тест по дисциплине" (реализовать проверку правильности для вопросов с элементами ввода типа RadioButton, ComboBox или однострочный текст) и вывода результатов проверки пользователю.

3. ИСХОДНЫЙ КОД

app/Core/View.php

```php
<?php

namespace App\Core;

use Xiaoler\Blade\Compilers\BladeCompiler;
use Xiaoler\Blade\Engines\CompilerEngine;
use Xiaoler\Blade\Factory;
use Xiaoler\Blade\FileViewFinder;

class View
{

    public static $path = ['app/views'];
    public static $cache = 'storage/views';
    private static $__instance;

    private $compiler;
    private $engine;
```

```
    private $finder;
    private $factory;

    private function __construct()
    {
        $this->compiler = new BladeCompiler(self::$cache);

        $this->engine = new CompilerEngine($this->compiler);
        $this->finder = new FileViewFinder(self::$path);

        $this->factory = new Factory($this->engine, $this->finder);
    }

    public static function register()
    {
        if (!function_exists('view')) {
            function view($view, $data = [], $_echo = true)
            {
                return View::view($view, $data, $_echo);
            }
        }
    }

    public static function view($view, $data = [], $_echo = true)
    {
        $result = self::instance()->factory->make($view, $data)->render();
        if ($_echo) {
            echo $result;
        }

        return $result;
    }

    private static function instance()
    {
        return self::$__instance ?? (self::$__instance = new self);
    }

    public static function share($key, $value = null)
    {
        self::instance()->factory->share($key, $value);
    }

}
```

app/Core/Validation.php

```
<?php

namespace App\Core;

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;
}

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];
        } else if (!empty($this->_messages[$field][$rule])) {
            if (!is_array($this->_messages[$field][$rule])) {
                $message = $this->_messages[$field][$rule];
            } else if (!empty($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], $tmp[0]]);
                } else if (is_int($tmp[0])) {
                    $count = [(int)$tmp[0], (int)$tmp[0]];
                }
            }
            if ($count[0] < 1) {
                return false;
            }
            $regex = sprintf('/([a-z0-9a-яё]+\s+){%d,%d}[a-z0-9a-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;
    }

}
```

app/Core/Router.php

```
<?php

namespace App\Core;

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, $len - 2);
        View::share([
            '_controller' => strtolower(self::$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 (!_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());
        }
    }
}
```

app/Controller/TestController.php

```
<?php

namespace App\Controller;

use App\Core\Controller;
use App\Core\Form;
use App\Validation\TestVerification;
use function App\Core\view;

class TestController extends Controller
{
    protected $_rules = [
        'name'           => 'required,words:2-3',
        'group'          => 'required',
        'points'         => 'required',
        'mark'           => 'required,int',
        'interpolation_' => 'count:1-3',
    ];
    protected $_messages = [
        'name'           => [
            'required' => 'Поле обязательное',
            'words'    => 'Введите в формате: Фамилия Имя Отчество',
        ],
        'group'          => 'Поле обязательное',
        'points'         => 'Поле обязательное',
        'interpolation_' => 'Должно быть выбрано 1-3 пунктов',
        'mark'           => [
            'required' => 'Поле обязательное',
            'int'      => 'Должно быть числом',
        ],
    ];

    function index()
    {
        $form = new Form($_REQUEST, TestVerification::run($this->_rules, $this->_messages));
        view('test.index', compact('form'));
    }

}
```

ВЫВОДЫ

В ходе лабораторной работы были повторены основы синтаксиса PHP. Было разработано Web приложение на PHP на базе методологии MVC. Написаны роутер, валидатор и контроллеры для функционирования системы.