**Часть 2. Практическая работа 8**

Архитектура MVC

В этой Практической работе вы:

* узнаете, что такое архитектура программы;
* познакомитесь с самым популярным шаблоном проектирования веб-приложений - MVC
* познакомитесь с реализацией этого шаблона в PHP на реальном примере

1. Архитектура программы

Архитектура - это базовая организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением, а также принципы, определяющие проектирование и развитие системы. [IEEE 1471]

Все значимые решения, направленные на организацию программы, а не на проблемы, решаемые с ее помощью, можно отнести к архитектуре.

Основа устойчивой архитектуры: строгое выделение уровней абстракции,

следование принципу минимальной связанности.

Что такое хороший код?

* Решающий проблему
* Простой
* Расширяемый
* Аккуратный

Шаблоны проектирования - набор типовых архитектурных решений.

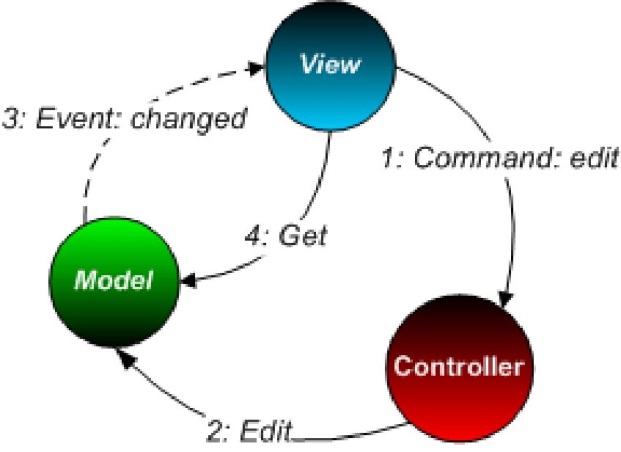
Как работает приложение:

* Обработка реакции пользователя (С)
* Взаимодействие с информационной моделью (M)
* Представление результата (V)

2. Архитектура MVC

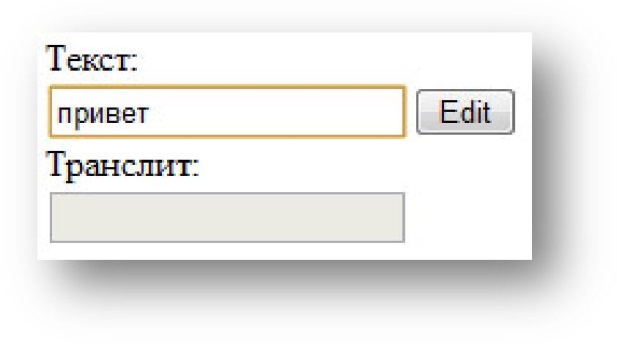
Паттерн MVC был разработан в 1978 г. и впервые применился при проектировании языка программирования Smalltalk в качестве модели для пользовательского интерфейса.

Классическая схема может выглядеть следующим образом:



На схеме показаны отношения между компонентами каркаса. Проиллюстрируем схемы небольшим примером.

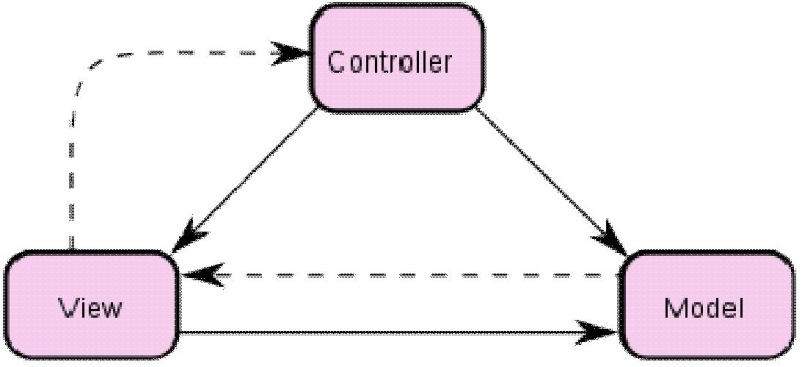
Представьте форму, где можно ввести текст, нажать на кнопку «Edit» и получить его транслитерацию.



Повторим шаги, изображенные на схеме:

1. Пользователь нажимает на кнопку «Edit», при этом представление (View) посылает сообщение контроллеру (Controller): «Command: edit».
2. Контроллер принимает сообщение и обращается к модели (Model), вызывая метод «Edit()».
3. В результате модель меняет свое состояние (запомненный в ней транслитерированный текст) и оповещает об этом представление: «Event: changed».
4. Представление принимает сигнал и обращается к модели за новым значением результата, вызывая ее метод «Get()».

Схема взаимодействия может быть и другой, например:



Но суть шаблона остается та же.

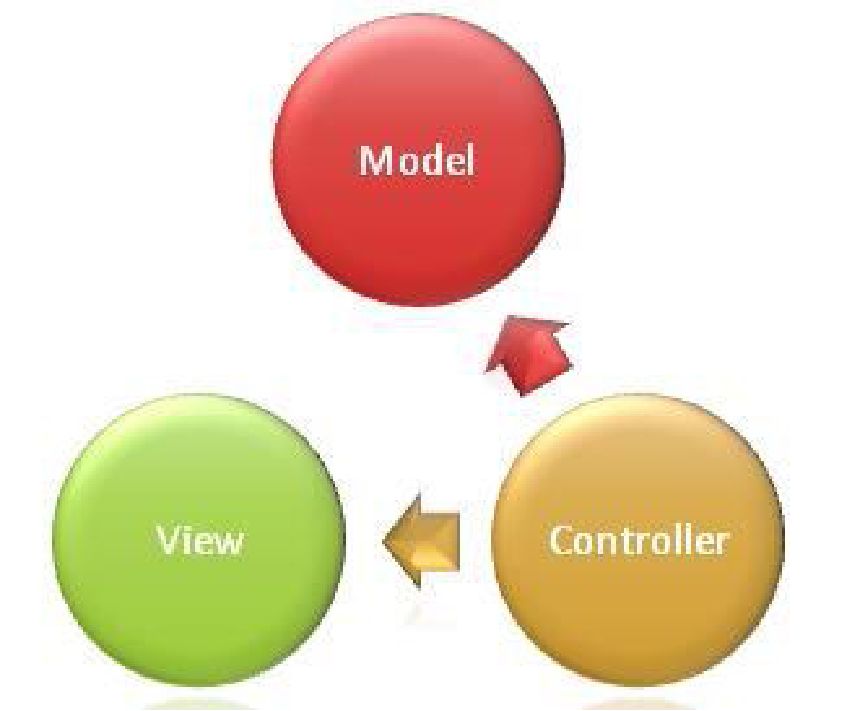
1. MVC в PHP

Контроллер - обработка запроса, проверка ввода, обращение к информационной модели, наполнение шаблона данными, вывод шаблона.

Модель - информационная модель приложения, отвечает за хранение информацию бизнес-логику процессов.

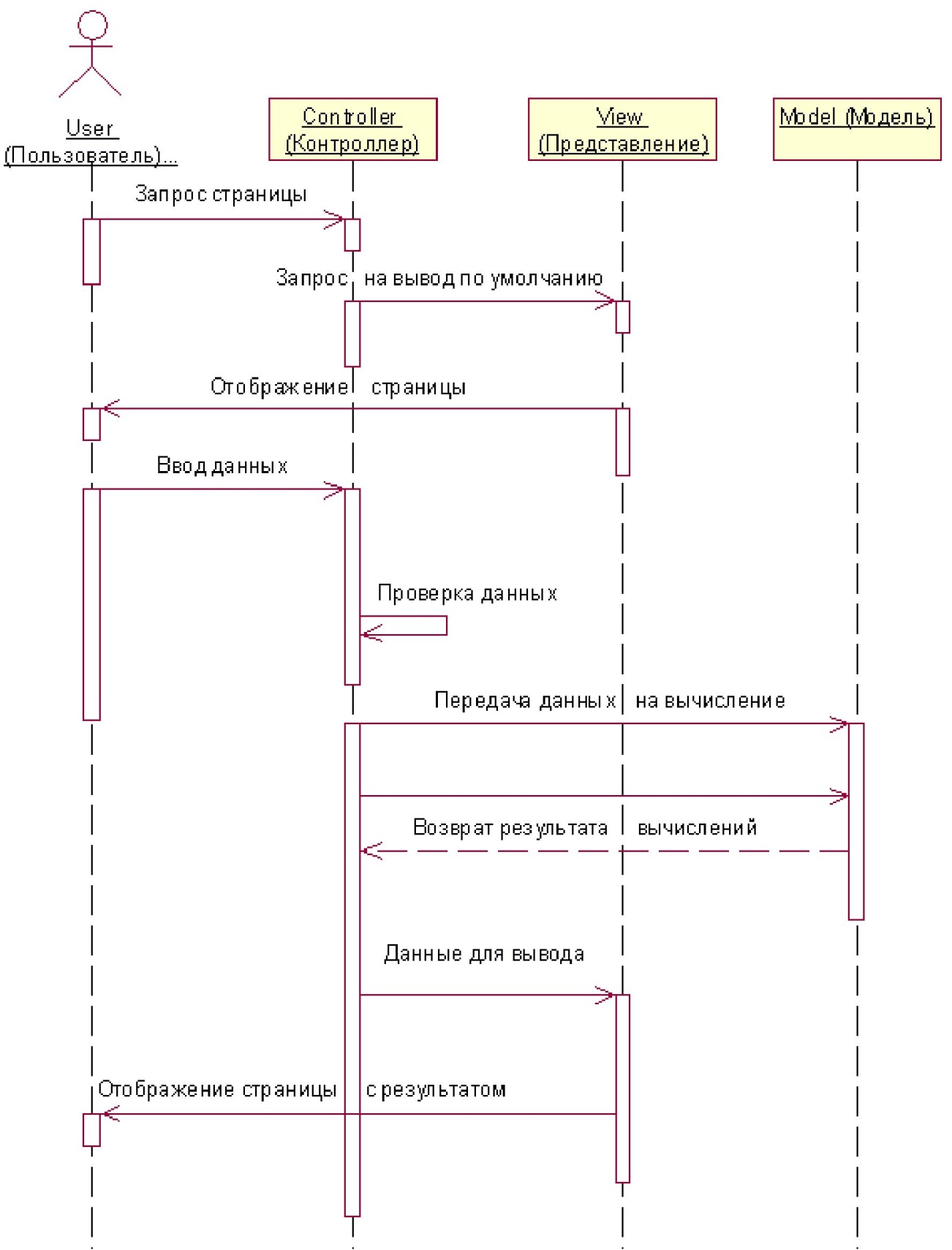
Представление - HTML шаблон.

Идеальная схема должна быть такой (руководствуясь принципом минимальной связанности):



Точка входа здесь - это контроллер.

Пример диаграммы последовательности для сайта, реализованного в рамках MVC:



1. Реализация MVC в PHP

Реализация MVC предполагает объектно-ориентированный подход. Однако шаблон проектирования - это всего лишь набор решений. Адаптируем их для PHP без применения ООП. Упрощение делается для того, чтобы сконцентрироваться на сути разделения логики, а также для того, чтобы материал смог применить читатель, не знакомый с ООП.

Рассмотрим снова пример с галереей фотографий. У нее есть два режима просмотра:

1. режим просмотра уменьшенных изображений (всех сразу)
2. режим просмотра фотографии полного размера (одной)

И возможность загружать фотографии на сервер. Также реализуем поддержку различных типов визуализации, чтобы оценить гибкость каркаса.

Вот так должен выглядеть наш сайт:

Галерея фотографий

**Таблица I Список**





Галерея фотографий



Просмотр фотографии

В программу будет две точки входа:

1. index.php (просмотр галереи)

**Назад**



1. photo.php (просмотр полноразмерной фотографии)

Эти два файла будем считать контроллерами.

В качестве модели будет выступать модуль, обеспечивающий работу с хранилищем изображений. Назовем его gallery.php и положим в папку model.

В роли представления будут выступать HTML шаблоны, они будут находиться в папке templates. Что такое шаблоны и для чего они нужны - будет видно дальше.

Страницы просмотра галереи и просмотра фотографии будут иметь общею шапку и низ страницы, отличаться будет только центральная часть.

Просмотр галереи будет иметь два типа визуализации:

1. в виде таблицы (по умолчанию)
2. в виде списка

Нам потребуется четыре шаблона:

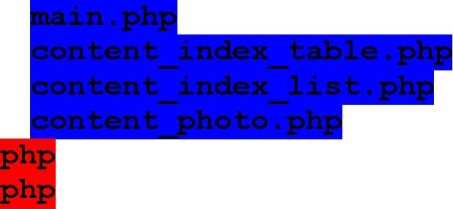
1. main.php (каркас страницы)
2. content\_index\_table.php (табличный вид содержимого галереи)
3. content\_index\_list.php (списочный вид содержимого галереи)
4. content\_photo.php (содержимое страницы просмотра фотографии)

Получается следующая структура сайта:

model/

gallery.php

templates/

index. photo.

Красным цветом обозначено представление, синим - контроллер, а зеленым - модель.

Модель

Начнем с реализации модели. Код приведен не полностью для минимизации и лучшей наглядности примера.

model/gallery.php:

<?php

// Функция возвращает объект фотографии (ассоциативный массив). function gallery item($id)

{ \_

// реализация пропущена // тип результата - array

}

// Функция возвращает список фотографий. function gallery list()

{ \_

// реализация пропущена // тип результата - array

}

// Функция возвращает путь к уменьшенному изображению. function gallery icon($photo)

{ \_

// реализация пропущена // тип результата - string

}

// Функция возвращает путь к полноразмерному изображению.

function gallery image($photo)

{ \_

// реализация пропущена

// тип результата - string

}

// Функция добавляет в галерею новую фотографию. function gallery add($file path, $file name)

{ \_ \_ \_

// реализация пропущена

}

?>

Мы определили лишь интерфейс модели, оставив реализацию пропущенной. Однако для примера реализации каркаса MVC она вовсе и не нужна.

Представление

Теперь рассмотрим шаблоны. Начнем с общего каркаса страницы.

templates/main.php: <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<**html**>

<**head**>

<meta content="text/html;charset=windows-1251" http-equiv="Content-Type"> <title><?=$title?></title>

</**head**>

<**body**>

<h1><?=$title?></h1>

<?php include $content; ?>

</**body**>

</ **html** >

Вас не должно смущать, что в шаблоне используются непонятно откуда взявшиеся переменные $title и $content. Их подставит контроллер. Но об этом позже.

<?=$title?> это сокращенный вариант записи

<?php echo $title; ?>

его удобно использовать в шаблонах. Также в шаблонах удобнее использовать альтернативные варианты записи конструкций if-else, foreach, for, while.

Выглядят они так:

if (<условие>):

<тело>

endif;

foreach ^инициализация цикла>):

<тело>

endforeach;

Остальные шаблоны будут подставляться в main.php таким образом:

<?php include $content; ?>

Ниже приведен их код.

templates/content\_index\_table.php:

<**br**/><**br**/>

<**table**>

<tr>

<?php $i = 0; ?>

<?php foreach ($photos as $photo): ?>

<?php if ($i % 3 == 2): ?>

</tr><tr>

<?php endif; ?>

< td>

<a href="photo.php?id=<?=$photo['id']?>">

<img src="<?=gallery icon($photo)?>" />

</a>

</ td>

<?php $i++; ?>

<?php **endforeach** ?>

</tr>

</**table**>

<br/><br/>

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

<input type="file" name="photo" />

<br/>

<input type="submit" уа1ие="Загрузить файл!" />

</**form**> **templates/content\_index\_list.php:**

<a href="index. р]ар">Таблица</а> | <Ь>Список</Ь> <br/><br/>

<?php foreach ($photos as $photo): ?>

<a href="photo.php?id=<?=$photo['id']?>">

<img src="<?=gallery icon($photo)?>" />

</a>

<br/>

<?php **endforeach** ?>

<br/><br/>

<form method="post" enctype="multipart/form-data"> <input type="file" name="photo" />

<br/>

<input type="submit" value="Загрузить файл!" />

</**form**>

templates/content\_photo.php:

<a href="index.php">Назад</a>

<br/><br/>

<img src="<?=gallery image($photo)?>" />

Контроллер

И, наконец, соберем все вместе, описав наши два контроллера. Их задача заключается в обработке запроса, выборе шаблона и подстановке нужных шаблону данных. Данные берутся, как правило, из модели.

Контроллер галереи загружает фотографию, если пользователь отправил файл. Иначе он извлекает из модели список фотографий, выбирает нужный шаблон (в зависимости от желания пользователя) и выводит этот шаблон, передав ему список фотографий.

index.php:

<?php

// Подключение библиотек. require\_once('model/gallery.php');

// Загружаем фотографию, если пользователь отправил файл. if (isset($ FILES['photo']))

{ \_

gallery add($ FILES['photo']['tmp name'], $ FILES['photo']['name']); header('Locaton: index.php');

**exit**();

}

// Подготовка данных.

$photos = gallery list();

// Заголовок страницы.

$title = 'Галерея фотографий';

// Выбор шаблона содержимого.

$content = ($ GET['view'] == 'list')

? 'templates/content index list.php'

: 'templates/content index table.php';

// Вывод HTML.

include 'templates/main.php';

?>

Контроллер просмотра фотографии еще проще.

photo.php:

<?php

// Подключение библиотек. require\_once('model/gallery.php');

// Подготовка данных.

$photo = gallery item($ GET['id']);

// Заголовок страницы.

$title = 'Просмотр фотографии';

// Выбор шаблона содержимого.

$content = 'templates/content photo.php';

// Вывод HTML.

include 'templates/main.php';

?>

Резюме

Старайтесь уделять достаточно внимания разработке архитектуры вашего приложения. Если изначально создать достаточно прочный и расширяемый каркас, усилия окупятся сторицей.

Для реализации модели MVC лучше подходит объектно-ориентированный подход. Существует множество готовых решений каркаса, например в Kohana, Zend Framework. Однако информации, изложенной в текущем Практическая работае, достаточно для того, чтобы понять архитектурные решения MVC и начать их использовать уже сейчас.

Ссылки

Для начинающих:

http:// chtivo. webho st .ru/articles/mvc.php

Есть смысл прочитать начало статьи: <http://lib.iuga.ru/article/view/163/1/68/>

Отличная статья на RSDN:

<http://www.rsdn.ru/article/patterns/generic-mvc.xml>

Задания:

Сделать миниблог на основе созданной БД и выданных вам исходников. Сайт должен соответствовать модели MVC.