С. В. Сапегин Ю.С. Скворцов Д.В. Романов

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СЕРВЕРНЫХ WEB (PHP 7, FRAMEWORKS)

Учебное пособие

Воронеж 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Воронежский государственный
технический университет»

С. В. Сапегин Ю.С. Скворцов Д.В. Романов

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СЕРВЕРНЫХ WEB (PHP 7, FRAMEWORKS)

Утверждено учебно-методическим советом университета в
качестве учебного пособия

Воронеж 2017

УДК 004.9
БКК

Сапегин С.В. Проектирование и разработка серверных Web (PHP 7, FRAMEWORKS): учеб. пособие / С.В. Сапегин, Ю.С Скворцов, Д.В. Романов. Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2017. 140 с.

Учебное пособие содержит теоретический и практический материал согласно программе дисциплины, а также контрольные вопросы и задания.

Предназначено для студентов 1 курса магистратуры.

Издание соответствует требованиям Федерального государственного образовательного стандарта высшего образования по направлению магистратуры 09.04.02 «Информационные системы и технологии» (направленности «Анализ и синтез информационных систем», «Разработка WEB-ориентированных информационных систем»), дисциплине «Проектирование и разработка серверных Web (PHP 7, FRAMEWORKS)».

Ил. 11. Библиогр.: 35 назв.

Рецензенты: кафедра программирования
и информационных технологий
Воронежского государственного
университета (зав. кафедрой канд.
физ.-мат. наук, проф. Н.А. Тюкачев);
д-р техн. наук, проф. В.Ф. Барабанов

ВВЕДЕНИЕ

Разработка Internet-приложений является на сегодняшний день одним из наиболее быстро развивающихся сегментов рынка. Необходимость использования глобальной сети для распространения информации, координации бизнес-процессов, управления различными проектными командами заставляет совершенно разные сферы бизнеса все больше обращать внимание на технологии и возможности, предоставляемые Internet и способные глобально изменить характер взаимодействия сотрудников в рамках одного предприятия.

Одним из наиболее популярных средств разработки Интернет-приложений является PHP. Изначально предоставляя существенные преимущества в процессе развертывания и поддержки Интернет-приложений, PHP вырос из легкого, полулюбительского средства написания логики к страницам HTML в мощную профессиональную технологию разработки, окруженную солидным стеком web-ориентированных средств. Использование PHP для реализации серверной логики в настоящее время является практически универсальным решением, охватывающим сегмент разработок от пользовательских мини-компонентов, встраиваемых в страницы HTML до мощнейших, высоконагруженных приложений, обрабатывающих десятки и сотни транзакций в секунду.

Настоящее пособие предназначено для использования в рамках курса “Разработка серверных приложений на PHP” и содержит в себе минимум информации, позволяющей составить сбалансированное впечатление об используемом в настоящее время стеке технологий, основанном на PHP, а также начать профессиональное развитие в этой области.

Пособие предназначено для студентов магистратуры направления 09.04.02 «Информационные системы и технологии», обучающихся по специализации «Разработка Web-ориентированных информационных систем».

1. АРХИТЕКТУРА INTERNET-ПРИЛОЖЕНИЙ

1.1. Среда работы Internet-приложений

В основу работы современных интернет-приложений в подавляющем большинстве случаев положен принцип архитектуры «клиент-сервер». Архитектура «клиент-сервер» описывает взаимодействие между компьютерами, при котором клиентская часть приложения запрашивает у сервера некоторые данные, а сервер отвечает на запрос. В случае интернет-приложений, средой выполнения клиентской части выступает Web-браузер, средой серверной логики – так называемые Web-серверы, приложения, расположенные в сети и отвечающие на запросы по протоколу HTTP.

Соответственно, само Web-приложение состоит из клиентской и серверной частей, взаимодействующих друг с другом на основе использования протокола HTTP. При этом, Web-приложения не зависят от операционных систем, используемых как на сервере, так и на клиенте.

Реализация пользовательского интерфейса, формирование запросов к серверу и обработка ответов от него происходят на клиентской части web-приложения. Серверная часть приложения предназначена для обработки запросов, хранения данных, реализации бизнес-логики и т.д. Результатом работы серверного приложения может быть как сформированная web-страница, предназначенная для просмотра на клиентской стороне с помощью браузера без дополнительной обработки, так и набор данных в формате XML, Json или иных форматах, позволяющих осуществлять их сериализацию для передачи и последующей обработки на клиенте.

Базовым протоколом для большинства интернет-приложений является протокол HTTP (HyperText Transfer Protocol — символьно-ориентированный клиент-серверный протокол прикладного уровня без сохранения состояния.

Основной задачей протокола является организация доступа к ресурсам сети, каждый из которых идентифицируется с помощью URI (Uniform Resource Identifier). В основном, ресурсами сети являются файлы различных типов. Протокол HTTP позволяет указать способ представления (кодирования) одного и того же ресурса в зависимости от его типа (mime-типа), языка и т.д. В настоящее время актуальна версия HTTP 1.1.

Протокол HTTP является текстовым и обладает структурой из трех частей, которые передаются в следующем порядке:

Стартовая строка (англ. Starting line) — определяет тип сообщения;

Заголовки — характеризуют тело сообщения, параметры передачи и прочие сведения;

Тело сообщения — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

С точки зрения характера взаимодействия через сеть, HTTP-сообщения можно разделить на HTTP-запросы (Request) и HTTP-ответы (Response). Примеры HTTP-запроса и ответа представлены ниже.

1) Обычный Get-запрос. Запрос-клиента:

GET /wiki/страница HTTP/1.1

Host: ru.wikipedia.org

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

(пустая строка)

Ответ сервера:

HTTP/1.1 200 OK

Date: Wed, 11 Feb 2009 11:20:59 GMT

Server: Apache

X-Powered-By: PHP/5.2.4-2ubuntu5wm1

Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT

Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close
(далее следует запрошенная страница в HTML).

2) Перенаправления. Предположим, что у вымышленной компании «Example Corp.» есть основной сайт по адресу «http://example.com» и домен-псевдоним «example.org». Клиент посылает запрос страницы «О компании» на вторичный домен:

GET /about.html HTTP/1.1
Host: example.org
User-Agent: MyLonelyBrowser/5.0

Так как домен «example.org» не является основным и компания не собирается в будущем его использовать в других целях, их сервер вернёт код для постоянного перенаправления, указав в заголовке Location целевой URL:

HTTP/1.x 301 Moved Permanently
Location: http://example.com/about.html#contacts
Date: Thu, 19 Feb 2009 11:08:01 GMT
Server: Apache/2.2.4
Content-Type: text/html; charset=windows-1251
Content-Length: 110
<html><body><ahref="http://example.com/about.html#contacts">Click here</body></html>

В заголовке Location можно указывать фрагменты как в данном примере. Браузер не указал фрагмент в запросе, так как его интересует весь документ. Но он автоматически прокрутит страницу до фрагмента «contacts», как только загрузит её. В тело ответа также был помещён короткий HTML-документ со ссылкой, с помощью которой посетитель попадёт на целевую страницу, если браузер не перейдёт на неё автоматически. Заголовок Content-Type содержит характеристики именно этого HTML-пояснения, а не документа, который находится по целевому URI.

Допустим, эта же компания «Example Corp.» имеет несколько региональных представительств по всему миру. И для каждого представительства у них есть сайт с соответствующим ccTLD. Запрос главной страницы основного сайта «example.com» может выглядеть так:

GET / HTTP/1.1
Host: example.com
User-Agent: MyLonelyBrowser/5.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru,en-us;q=0.7,en;q=0.3
Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.7

Сервер принял во внимание заголовок Accept-Language и сформировал ответ со временным перенаправлением на российский сервер «example.ru», указав его адрес в заголовке Location:

HTTP/1.x 302 Found
Location: http://example.ru/
Cache-Control: private
Date: Thu, 19 Feb 2009 11:08:01 GMT
Server: Apache/2.2.6
Content-Type: text/html; charset=windows-1251
Content-Length: 82
(пустая строка)
<html><body><ahref="http://example.ru">Example
Corp.</body></html>

Обратите внимание на заголовок Cache-Control: значение «private» сообщает остальным серверам (в первую очередь прокси) что ответ может кэшироваться только на стороне клиента. В противном случае не исключено, что следующие посетители из других стран будут переходить всё время не в своё представительство.

Согласно принципам работы протокола HTTP, основной моделью взаимодействия приложений в среде Интернет является модель «клиент-сервер». При этом, клиент

осуществляет запрос доступа к различным ресурсам на основе HTTP, а сервер обрабатывает его запросы и выдает ответы. Следует заметить, что HTTP по своему характеру является **синхронным** протоколом, т.е. клиент, отослав запрос, ожидает ответ сервера. Для преодоления ограничения синхронности (1 запрос – 1 ответ) в современных браузерах реализованы следующие технологии:

1. Конвейер HTTP (HTTP pipelining) – технология упорядоченной отсылки нескольких запросов одновременно – причем ответы на эти запросы должны приходить в таком же порядке. Технология существует в виде опции, позволяющей увеличить доступность ресурсов для категорий пользователей с большим временем отклика.
2. Технология AJAX (Asynchronous Javascript and XML), позволяющая конструировать и выполнять запросы в фоновом режиме браузера, не мешая основному потоку событий.

Большинство информации, находящейся в Интернет, представлено в формате HTML (HyperText Markup Language), актуальная версия которого на данный момент – HTML5. Язык HTML — это язык разметки, которая заключается в том, что в обычный текст добавляются специальные команды HTML, описывающие, как должен выглядеть т восприниматься данный текст. С помощью команд HTML можно внедрять в документы различные ресурсы сети (изображения, видеоролики, программные компоненты), а также задавать правила их отображения. Большинство гипертекстовых документов в Интернет связаны в общую сеть в соответствии с принципами WWW при помощи гиперссылок – частей гипертекстового документа, ссылающихся на другие элементы документа, либо на ресурсы в сети.

WWW (World Wide Web), или Всемирная Паутина была создана в 1989 году британским ученым Тимом Бернерсом-Ли. Он предложил разработать «гипертекстовую систему» для облегчения возможности обмена информацией между

людьми. Эта система позволяла создавать документы, содержащие ссылки на другие документы, обрабатывать их и перемещать пользователя из одного документа в другой, а также связывать всю информацию в одну огромную связанную систему.

С появлением Всемирной Паутины стало возможным представлять информацию в естественной форме текста, рисунков, анимации, звука, видео, а также небольших программ, которые делают страницу интерактивной, то есть изменяющейся в зависимости от действий пользователя.

Служба WWW – это служба поиска и просмотра гипертекстовых документов. Такие документы называются Web-страницами. Web-сайт (Web-узел) – совокупность хранящихся вместе Web-страниц.

WWW основывается на системе ссылок, которые содержат адреса необходимых документов. Для обозначения адресов ресурсов Интернета используется адресная система URL (унифицированный указатель ресурсов). Она позволяет указывать имя документа и протокол доступа к нему.

Форма записи URL следующая:

протокол://домен:порт/каталог/файл

При этом использование пробелов не допускается.

При обращении к адресу URL обработка запроса осуществляется в 4 фазы:

- connection;
- request;
- response;
- close.

На первой фазе соединения (connection) Web-клиент устанавливает соединение с сервером. Вторая фаза – запрос (request). При успешном соединении клиент посылает на сервер запрос о том, какой объект нужно найти. В него включена команда для сервера (например, GET – взять объект). Следующая фаза – ответ (responce), на ней сервер посылает ответ клиенту. На завершающей фазе закрытия

(close) соединение отключается, в браузере отображаются полученные данные.

1.2. Клиентские технологии Web-приложений

Средой работы клиентской части Web-приложения является web-браузер – специальная программа, осуществляющая взаимодействие с Web-серверами и представляющая полученную информацию. При этом, для задания правил отображения информации используется HTML (Hyper Text Markup Language) — стандартизированный язык разметки гипертекстовых документов, который используется для разметки большинства страниц в сети Internet. Он представляет собой текст, содержащий управляющие коды (теги). Эти теги интерпретируются браузером, а результат отображается на экране. Теги заключаются в угловые скобки < >. Существуют парные и одиночные теги. Парные состоят из открывающего и закрывающего тегов. Если тег закрывающий, перед его именем в угловых скобках ставится «/». Например:

<body>
…
</body>

Теги могут иметь атрибуты. Атрибуты бывают необязательными и обязательными.

Для реализации динамического содержимого клиентских страниц Web-приложений используется язык Javascript, разработанный в 1995 году. Несмотря на то, что в настоящее время существует целый стек технологий на основе Javascript, позволяющий вести разработку без привлечения других языков, основная масса Web-приложений в сети использует Javascript именно для реализации клиентской логики. Код Javascript интегрирован с HTML и добавляется на web-страницу обычно при помощи парного html-тега <script> .

Основные возможности Javascript:

- создание новых тегов и удаление существующих, изменение стилей элементов;
- ответ на действия пользователя выполнением функций;
- отправление запросов на сервер, в т.ч. ;на основе технологии Ajax
- работа с DOM и его окружением, включая cookie, браузерные расширения и т.д.

1.3. PHP как серверный язык

PHP в настоящее время является одним из наиболее распространенных серверных языков веб-программирования. Начавшись в 1994 году, как скриптовый язык для создания персональных страниц (Personal Homepage Preprocessor – PHP), PHP эволюционировал в мощное средство разработки промышленных Интернет-приложений, включая сегмент Highload, обладающее собственной экосистемой средств разработки, профилирования, запуска и мониторинга программных систем. Низкий порог вхождения и относительная простота разработки и развертывания PHP-приложений, наряду с высокопроизводительной архитектурой (особенно, в PHP7) позволяют использовать этот язык максимально широко – от разработки единичных страничек уровня «Hello world!» до систем класса Enterprise, разворачиваемых в облаке, либо с использованием пула высокопроизводительных серверов. Практика успешного использования PHP в крупных проектах даже привела к тому, что PHP в настоящее время позиционируется, как один из языков платформы WebSphere – крупнейшей в мире интеграционной платформы корпорации IBM, используемой в банковском и финансовом секторе, а также для автоматизации крупных промышленных предприятий.

1. PHP 3. PHP 3.0 можно считать первой версией языка PHP. Официально PHP 3.0 был выпущен в 1998 году. В нем

имелась возможность расширения ядра, что считалось его преимуществом. Так же PHP 3.0 поддерживал ООП синтаксис. Язык достаточно быстро набрал популярность и привлек сторонних разработчиков.

2. PHP 4. В 2000 году был представлен PHP 4. Язык был основан на новом движке «ZendEngine». Появились дополнительные функции, производительность значительно повысилась. Кроме этого, была включена поддержка большего числа серверов, HTTP сессий, безопасности способов обработки данных, новых языковых конструкций и буферизации вывода.

3. PHP 5. В июле 2004 года был выпущен PHP 5. Язык работал под управлением нового ядра «ZendEngine 2.0», которое имело отличную объектную модель и массу иных новшеств.

В 2009 появилась версия 5.3. Основными нововведениями являются:

- динамический доступ к статическим методам;
- возможность объявления константы вне класса;
- использование Heredoc (определение строковых переменных);
- поддержка пространств имен;
- замыкания;
- позднее статическое связывание и др.

4. PHP 5.4 – релиз состоялся в 2012 году. Преимущества данной версии:

- трейты (механизм повторного использования кода);
- объявление массива с кратким синтаксисом;
- отсутствие функции register_globals (глобальных переменных).

PHP 5.5 был представлен в 2013 году. В этой версии появились:

- возможность обращение к символам как к элементам массива;
- генераторы;
- блок finally (try/catch) – обработка исключений;
- расширение описания массивов.

5. PHP 7. В настоящее время последней версией языка является PHP 7 (2015 год). В новой версии улучшена производительность, а также добавлены новые возможности:

- сокращенная конструкция use;
- NULL-коалесцирующий оператор;
- комбинированный оператор сравнения;
- анонимные классы;
- **метод Closure::call() – анонимные функции;**
- уменьшение используемой памяти;
- классы исключений –Exception, Error, и др.

Интеграция PHP с HTML

Для интеграции PHP-кода в HTML-страницу используется тег <?php …?>. Все, что находится вне этого тега, игнорируется интерпретатором PHP. В этом случае PHP-код может содержаться в HTML-документе. Интерпретатор PHP отображает все, что находится после ?>.

Помимо тега <?php …?> существует и другие варианты обозначения PHP-кода.

Интеграция PHP с Javascript

Написанные на PHP скрипты выполняются на сервере, в то время как скрипты на JavaScript выполняются на стороне пользователя. Для обмена данными между клиентом и сервером используются следующие варианты:

- использование cookie;
- использование Ajax;
- изменения текста скрипта.

1.4. Дополнительные средства разработки Web-приложений

Современные Web-приложения представляют собой достаточно сложные, трудоемкие, многокомпонентные системы, разрабатываемые целыми коллективами различных специалистов. Этому способствует широкое распространение наиболее полезных и популярных приложений, минуя границы отдельных государств и целых континентов. Для разработки таких приложений существует бесчисленное множество различных средств, технологий, фреймворков, решений и т.д. Рассмотрим вкратце некоторые, наиболее популярные библиотеки и инструменты, в настоящее время обладающие статусом «стандарт де-факто» для индустрии разработки Web-приложений.

TwitterBootstrap – фреймворк, который был создан разработчиками Twitter. В настоящее время TwitterBootstrap является одним из самых популярных фреймворков. При помощи шаблонов в TwitterBootstrap можно изменять уже модифицированные элементы. Подключение шаблонов происходит добавлением вызова CSS-шаблона.

Основные средства TwitterBootstrap:

- тип макета;
- формы;
- навигация;
- таблицы;
- медиа;
- модульная сетка шаблона;
- типографика и др.

.

Этот фреймворк начал разрабатываться как внутренняя библиотека компании Twitter под названием Twitter Blueprint. После нескольких месяцев разработки он был открыт под названием Bootstrap 19 августа 2011 года. Основными

нововведениями второй версии, появившейся 31 января 2012 года, стали 12-колоночная сетка и поддержка адаптивности.

Третья версия выпущена 19 августа 2013 года. В ней адаптивность получила дальнейшее развитие, был осуществлён переход к концепции mobile first, оптимизации прежде всего под мобильные устройства. Дизайн по умолчанию стал плоским. Работа над четвёртой версией начата 29 октября 2014 года. Альфа версия вышла 19 августа 2015 года. Первая бета версия выпущена 10 августа 2017.

Способы установки:

1) вручную, посредством самостоятельного скачивания этой платформы (архива) с сайта http://getbootstrap.com и библиотеки jQuery. Данный способ является наиболее трудоёмким, так как Вам придётся выполнять все действия связанные с установкой и последующим обновлением платформы вручную;

2) с помощью указания файлов, находящихся на серверах CDN. Данный способ имеет некоторые преимущества, перед предыдущим способом, а именно:

уменьшает нагрузку на Ваш сервер, т.к. загрузка файлов происходит с CDN;

увеличивает скорость загрузки веб-страницы, т.к. необходимые файлы берутся из кэша браузера. Это происходит только в том случае, если до этого (т.е. на других сайтах) эти файлы загружались аналогичным образом, т.е. из этого же CDN;

3) с помощью Bower (http://bower.io);

4) с помощью npm (http://www.npmjs.com).

FontAwesome – коллекция иконок (всего их 439). Библиотека FontAwesome абсолютно бесплатна. Использовать ее можно в любом фреймворке, хотя изначально она была создана для TwitterBootstrap.

Для использования FontAwesome необходимо добавить CSS-файл и папку с сгенерированными шрифтами в папку с проектом, после чего добавить CSS-файл в HTML-документ.

JQuery - библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. Сейчас разработка jQuery ведётся командой jQuery во главе с Джоном Резигом.

Возможности

1. Движок кросс-браузерных CSS-селекторов Sizzle, выделившийся в отдельный проект;
2. Переход по дереву DOM, включая поддержку XPath как плагина;
3. События;
4. Визуальные эффекты;
5. AJAX-дополнения;
6. JavaScript-плагины.

1.5. Лабораторная работа №1. Быстрое прототипирование Internet-приложений

1.5.1. Постановка задачи

Индустрия разработки Web-приложений является одной из наиболее динамичных отраслей мировой экономики, в которой конкурируют разработчики, команды и решения со всех концов мира. Жесткая, глобальная конкуренция в этой сфере стимулирует постоянное изучение, внедрение и развитие самых передовых технологий и методологий разработки.

Одним из полезных принципов разработки ПО, способным обеспечить конкурентные преимущества, является быстрое прототипирование. Смысл быстрого прототипирования заключается в создании макета приложения для оценки предлагаемых концепций и решений, выявления требований заказчика на ранних стадиях и оценки характера

дальнейшей разработки ПО. Целью лабораторной работы является построение прототипа программного продукта с использованием инструментов Twitter Bootstrap и Font Awesome. Прототип должен имитировать наиболее ключевую, с Вашей точки зрения, функциональность проектируемого приложения и быть понятным для предполагаемого заказчика. Прототип должен быть выполнен в виде набора HTML-страниц, связанных гиперссылками и оформленных при помощи вышеперечисленных инструментов. Использование аналогов вместо Twitter Bootstrap и Font Awesome обоснованно допускается.

Критериями оценки прототипа являются:

1. Предполагаемое сходство с реальными приложениями
2. Использование реальных данных и реальных ситуаций
3. Удобство работы с прототипом для конечного пользователя
4. Понятность и близость к предметной области

Примерный ход выполнения лабораторной работы

1. Выбор темы приложения и анализ предметной области. Допустим, целью прототипирования является приложение для организации научно-учебной деятельности. Производим краткий анализ предметной области, а также анализ пользовательских требований и пожеланий именно того самого предполагаемого пользователя, для которого планируется разработка прототипа. В нашем случае это будет сетевой инженер, в обязанности которого входит обслуживание, контроль и учет оборудования нескольких лабораторий.

2. Определяем основные обязанности выбранного пользователя. В случае, если пользователь в процессе работы формирует некоторые бумажные документы – обязательно снимаем с них копии. Лучшим решением при прототипировании является, по возможности, наиболее полное копирование структуры бумажных документов.

В нашем случае это:

- подсчет количества различного оборудования
- его инвентаризация
- учет срока службы каждой единицы оборудования, регистрация инцидентов
- плановая и экстренная замена оборудования в случае поломки
- ведение финансовой отчетности

3. Проектируем общую структуру приложения, разбиваем его на модули. В нашем случае один из вариантов декомпозиции приложения на модули будет выглядеть следующим образом:

- Планирование
- Операционный учет
- Отчетность

Эти пункты будут входить в главное меню прототипа наряду с набором следующих сервисных пунктов:

- Пользователь
- Справочники

4. Создаем папку проекта, располагаем внутри Twitter Bootstrap и Fontawesome для последующего подключения, создаем страницу index.html, которая будет являться главной страницей приложения.

Исходный код index.html:

```
      <div class="container-fluid">
      <div class="navbar-header">
      <button type="button" class="navbar-toggle
collapsed" data-toggle="collapse" data-
target="#navbar-main">
      <span class="icon-bar"></span>
      <span class="icon-bar"></span>
       <span class="icon-bar"></span>
          </button>
            <a class="navbar-brand"
href="eq_count.html">Планирование</a>
```

```
        <a class="navbar-brand"
href="in_developing.html">Операционный учет</a>
        <a class="navbar-brand"
href="in_developing.html">Отчетность</a>
        <a class="navbar-brand"
href="in_developing.html">Пользователь</a>
        <a class="navbar-brand"
href="in_developing.html">Справочники</a>
        </div>
        <div class="collapse navbar-collapse"
id="navbar-main">
        </div>
      </div>
```

5. Определяем внешний вид формы «подсчет количества различного оборудования» и создаем ее в файле eq_count.html.

Исходный код eq_count.html:

```
<table class="table">
 <thead>
 <tr>
 <th>Наименование оборудования</th>
 <th>Годовая трудоемкость</th>
 <th>Количество оборудования, шт</th>
  <th>Коэффицент загрузки</th>
 </tr>
</tbody>
 </table>
```

6. Наполняем форму тестовыми данными. Данные необходимо брать либо из реальных документов, либо из подходящих по смыслу документов, найденных в сети Интернет. Заполнение полей формы данными вида «тест», «привет мир» является наиболее худшим вариантов, ставящим под сомнение саму идею прототипирования (рис 1).

```
<table class="table ">
 <thead>
 <tr>
 <th>Наименование оборудования</th>
```

```
<th>Годовая трудоемкость</th>
<th>Количество оборудования, шт</th>
<th>Коэффицент загрузки</th>
</tr>
</thead>
<tbody>
<tr>
<td>Пресс</td>
<td>143450</td>
<td>36</td>
<td>0.99</td>
</tr>
<tr>
<td>Ковачная машина</td>
<td>27660</td>
<td>18</td>
<td>0.98</td>
</tr>
<tr>
<td>Фрезерный станок</td>
<td>96550</td>
<td>29</td>
<td>0.95</td>
</tr>
</tbody>
</table>
```

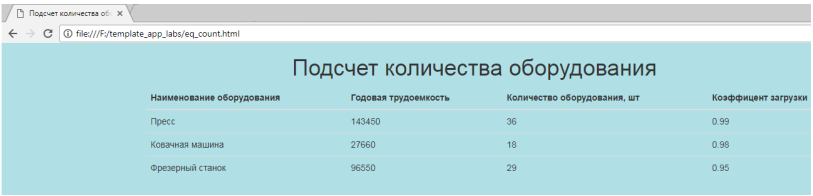

Рис 1.1. Подсчет количества оборудования

7. Настраиваем гиперссылки и формируем пакет. Неработающие ссылки должны указывать на страницу «в разработке».

```
     <aclass="navbar-brand"
href="eq_count.html">Планирование</a>
     <a class="navbar-brand"
href="in_developing.html">Операционный учет</a>
     <a class="navbar-brand"
href="in_developing.html">Отчетность</a>
     <a class="navbar-brand"
href="in_developing.html">Пользователь</a>
     <a class="navbar-brand"
href="in_developing.html">Справочники</a>
```

1.6. Контрольные задания

1.6.1. Варианты заданий для реализации

1. Приложение «Бухгалтерия-онлайн»
2. Личный кабинет клиента банка
3. Приложение кредитного инспектора
4. Приложение для налоговой службы
5. Рабочее место преподавателя ВУЗа
6. Приложение для студента
7. Приложение для туриста
8. Интернет-представительство гостиницы
9. Приложение для управления производством мебели
10. Приложение для автоматизации сельского хозяйства
11. Приложение для управления слесарным цехом
12. Приложение вагоноремонтного депо
13. Библиотека финансовых инструментов для маклера
14. Приложение ЦУП космодрома
15. Система управления аэропортом

1.6.2. Вопросы для самопроверки

1. Что такое Web-приложение?
2. Из каких частей состоит Web-приложение?

3. Служба WWW.
4. Адресная система URL.
5. Протокол HTTP.
6. Язык HTML.
7. Язык JavaScript.
8. Серверный язык PHP, версии.
9. Интеграция PHP с другими языками.
10. Что такое шаблоны и шаблонизаторы?
11. TwitterBootstrap.
12. FontAwesome .

2. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА PHP

2.1. Web-серверы. Apache

Веб-сервер – сервер, который принимает HTTP-запросы от клиентов и выдающий им HTTP-ответы вместе с HTML-страницей или другими данными. Клиентами обычно являются веб-браузеры.

Apache является веб-сервером с открытым исходным кодом. Он поддерживает такие операционные системы, как Microsoft Windows, Linux, BSD, Mac OS, BSD, NovellNetWare. Сервер прост в установке и настройке.

Достоинства Apache:

- открытый исходный код;
- бесплатная лицензия;
- кроссплатформенность;
- надежность;
- гибкость конфигурации.

Сервер Apache можно настраивать при помощи конфигурационных файлов. Если стандартная функциональность не устраивает пользователя, есть возможность добавлять распространяемые модули.

Основная функция Apache – обслуживание HTML-сайта. Сервер получает запросы и отправляет ответы браузеру. Также Apache позволяет разграничивать доступ к страницам сайта для пользователей.

1. Nginx – веб-сервер, почтовый прокси-сервер, написанный русским разработчиком И. Сысоевым. Сервер работает на ОС Unix.

Сервер Nginx применяется для статических веб-сайтов. Также Nginx используют как прокси-сервера перед динамическими сайтами.

Основные возможности:

- обслуживание статические запросы;

- акселерированное проксирование без кэширования;
- поддержка серверов с кэшированием;
- модульность;
- поддержка SSL;
- поддержка HTTP/2.

2.2. СУБД

Система управления базами данных (СУБД) – совокупность программных и языковых средств, обеспечивающих создание и использование реляционных БД.

1. MySQL – самая распространенная СУБД. MySQL обладает высокой скоростью работы. Это достаточно надежная и гибкая СУБД. В поставку PHP включена поддержка сервера MySQL. MySQL является бесплатной СУБД.

Особенности MySQL:

- система с открытым исходным кодом;
- сервер MySQL быстрый, надежный. Он работает в строенных и клиент-серверных системах;
- содержит множество типов таблиц;
- высокая степень масштабируемости;
- размер файла ограничивается возможностями используемой ОС.

OracleDatabase – объектно-реляционная СУБД, созданная компанией Oracle. СУБД поддерживает технологии, позволяющие использовать объектно-ориентированный подход. Сервер Oracle позволяет хранить и обрабатывать различные типы данных, создавать новые типы данных и методы доступа к ним.

Особенности OracleDatabase:

- одновременный доступ множества пользователей к одним и тем же данным;

- реализация устойчивого чтения;
- механизм блокировок;
- транзакции, не меняющие данные;
- портируемость.

2. PostgreSQL – свободно распространяемая объектно-реляционная СУБД. Эта система использует индексы, что повышает ее производительность. Также PostgreSQL содержит тонкую систему блокировок и систему управления буферами памяти.

Особенности PostgreSQL:

- поддержка CTE;
- поддержка оконных функций;
- поддержка schema;
- масштабируемость;
- кроссплатформенность;
- группировка по именным полям.

3. MongoDB – документоориентированная СУБД с открытым исходным кодом. Запросы в MongoDB пишутся на BSON. Данная СУБД используется в случаях, если таблицы можно представить в виде объектов.

Особенности MongoDB:

- динамические запросы;
- документоориентированное хранение;
- поддержка индексов;
- большой набор операций над данными;
- профилирование запросов.

4. Redis – нереляционная высокопроизводительная СУБД с открытым исходным кодом. Данная СУБД хранит данные, доступ к которым осуществляется по ключу. Redis позволяет сохранять данные на диск. В СУБД существует возможность хранения массивов, словарей, множества без повторов.

Особенности Redis:

- однопоточный сервер;
- репликация;

- поддержка транзакций;
- поддержка пакетной обработки команд;
- механизм publish/subscribe;
- возможность защитить доступ к серверу паролем.

5. Composer – менеджер зависимостей для PHP. Он позволяет легко устанавливать различные пакеты. При указании необходимых для проекта библиотек Composer самостоятельно их установит.

Composer скачивает пакеты и их зависимости. Все пакеты устанавливаются в текущую директорию, после чего генерируется файл autoload.php. С помощью этого файла можно подключить установленные библиотеки в коде проекта.

6. Git – распределенная система управления версиями файлов. Системы управления версиями файлов регистрируют изменения в одном или нескольких файлах для того, чтобы в дальнейшем была возможность вернуться к более старым версиям этих файлов. Главное отличие Git заключается в том, что эта система иначе смотрит на свои данные.

Система Git является распределенной системой контроля версий. Это означает, что клиенты не просто выгружают последние версии файлов, а полностью копируют весь репозиторий. В таких системах можно одновременно работать по-разному с разными группами людей в рамках одного проекта. Набор версий может быть разделен между разными хранилищами.

Git считается самой быстрой распределенной системой, при этом она использует самое компактное хранилище ревизий.

Такие проекты как Ядро Linux, PHP, Qt, Android, jQuery используют систему Git.

7. TDD (разработка через тестирование) – подход к разработке программного обеспечения, основанный на повторении коротких циклов разработки. Такие циклы включают в себя написание теста, покрывающего желаемое

изменение, написание кода, позволяющего пройти тест, и рефакториинг нового кода.

Тест – процедура, проверяющая работоспособность кода. При TDD программист должен создавать модульные тесты. Эти тесты определяют требования к коду перед тем, как он будет написан. Для тестирования часто используют специальные библиотеки. С помощью них программист создает и запускает набор тестов.

Разработка через тестирование обладает рядом преимуществ. Одним из них является то, что помимо проверки работоспособности кода, такой подход влияет и на дизайн программы. Хоть при TDD и приходится писать больший объем кода, время, затраченное на разработку программы, обычно оказывается меньше. Это происходит за счет уменьшения количества ошибок. Разработка через тестирование делает код более гибким и расширяемым. Также тесты можно использовать в качестве документации, т.к. они наглядно демонстрируют использование тестируемого кода.

Разработка программного обеспечения через TDD обычно состоит из следующих шагов:

1. Пишется и запускается тест. После запуска должна появиться ошибка, что такого класса/функции не существует.
2. Создается класс, пустой метод и снова запускается тест. В этом случае появляется ошибка, что тестируемый метод возвращает неправильное значение.
3. Пишется минимальная реализация метода для прохождения теста. После запуска должно появиться сообщение об успешном прохождении теста.
4. Добавляются тесты с разными входными и ожидаемыми значениями. После прохождения этих тестов появится ошибка, что результат метода не совпадает с новыми значениями.
5. Метод полностью переписывается для прохождения всех тестов.

2.3. Средства разработки

1. jEdit – бесплатный кроссплатформенный редактор с открытым исходным кодом. jEdit написан jEdit на языке Java. Редактор предназначен для программистов. Существует возможность расширения функциональности jEdit с помощью макросов, написанных на скриптовых языках.

В редакторе существуют авто отступ и подсвечивания синтаксиса более 200 языков. jEdit поддерживает множество символьных кодировок.

Основные возможности редактора:

- клавиатурные сокращения для команд;
- online помощь;
- неограниченное количество операций отмены и повтора;
- неограниченное число буферов обмена;
- запоминание удаленного текста;
- команды для работы со словами, строками и параграфами;
- использование маркеров для сохранения позиция в файле;
- неограниченное число открытых окон редактирования;
- возможность выделения прямоугольных областей;
- перенос слов.

2. RJ TextEd – бесплатный текстовый редактор для программистов, работающий в операционной системе Microsoft Windiws. Написан RJ TextEd на языке ObjectPascal. RJ TextEd обладает расширенными функциями для работы с исходным кодом. Редактор также полезен при создании сайтов. В RJ TextEd поддерживается работа с HTML и CSS. Существует возможность работы с множеством документов с помощью кладочного режима. Для упрощения работы в

редактор встроены шаблоны. Для комментирования текста в редакторе используется блок комментариев. Для файлов HTML, PHP, ASP реализован предварительный просмотр.

Основные возможности редактора:

- сворачивание кода;
- шаблоны;
- выделение текста блоками;
- интеграция в проводник Windows;
- темы оформления;
- подсветка синтаксиса;
- автоматическое завершение;
- многоуровневый откат и повтор действий;
- конвертирование файлов и др.

3. KomodoEdit – бесплатный кроссплатформенный редактор для динамических языков программирования. KomodoEdit построен на движке Mozilla и поддерживает множество языков. Редактор имеет проработанный удобный интерфейс. Имеется возможность настройки стиля интерфейса. В программе поддерживается полноэкранный режим. В редактор встроена библиотека, содержащая большое количество элементов для автоматической вставки участков кода.

В KomodoEdit реализована подсветка синтаксиса и автоисправление ошибок. Для основных языков программирования есть специальные настройки отображения. В программе удобно реализован поиск. Можно искать не только по текущему документу, но и по всем открытым во вкладках файлах. Для быстрого исправления ошибок есть функции «Найти и заменить»

Редактор KomodoEdit имеет систему поддержки дополнений, с помощью которых можно настраивать интерфейс, добавлять новые функции и дополнительно включать совместимость с языками программирования и фреймворками.

4. NetBeans IDE – бесплатная интегрированная среда разработки приложений для программистов. NetBeans IDE позволяет разрабатывать программы на языках Java, PHP, C, C++, Python и др.

Рабочую область среды можно настроить. Существует возможность расширения функций редактора. NetBeans позволяет генерировать и вставлять в исходный код программы стандартные фрагменты кода на различных языках. Браузер классов позволяет посмотреть иерархию и структуру классов Java. Во время ввода производится проверка на ошибки. При создании группы объектов существует возможность быстро открывать и закрывать несколько сгруппированных проектов одновременно.

Среда NetBeans IDE подходит не только для опытных программистов. Подробная встроенная справочная система обеспечит быстрый старт для начинающих разработчиков.

5. OpenServer — это портативная серверная платформа и программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.

Программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администрированию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

Хотя изначально программные продукты, входящие в состав комплекса, не разрабатывались специально для работы друг с другом, такая связка стала весьма популярной среди пользователей Windows, в первую очередь из-за того, что они получали бесплатный комплекс программ с надежностью на уровне Linux серверов.

6. XAMPP - кроссплатформенная сборка веб-сервера (развитие LAMP), содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое

количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер.

Полный пакет содержит:

- Web-сервер Apache с поддержкой SSL
- СУБД MySQL;
- PHP;
- Perl;
- FTP-клиент FileZilla;
- POP3/SMTP сервер;
- утилиту phpMyAdmin.

7. Denwer - набор дистрибутивов (локальный сервер WAMP) и программная оболочка, предназначенные для создания и отладки сайтов на локальном ПК под управлением ОС Windows. Сразу после установки доступен полностью работающий веб-сервер Apache, работающий на локальном компьютере, на котором может работать неограниченное количество сайтов, что очень эффективно для разработки и отладки сценариев PHP без загрузки его файлов на удаленный сервер. Для запуска практически всех утилит «Денвера» используется приложение Run в подкаталоге /denwer (или /etc) корневого каталога установки «Денвера». При запуске создается виртуальный диск (по умолчанию Z:), где хранятся все файлы проектов. Третья версия поддерживает работу со съемного флеш-накопителя. Особенностью, отличающей Denwer от других WAMP-дистрибутивов, является автоматическая правка системного файла hosts, являющегося локальным аналогом DNS-сервера, что позволяет обращаться к локальным сайтам, работающим под управлением Денвера, по именам, совпадающим с именем папки, расположенной в каталоге home Денвера.

2.4. Лабораторная работа №2. Генерация файлов с помощью библиотеки GD

2.4.1. Постановка задачи

PHP не только идеально подходит для вывода HTML страниц, но также включает в себя мощные средства создания изображений "на лету". Приведем несколько примеров:

1. Создание кнопок с надписями, текст которых хранится в базе данных
2. Графическое отображение статистических данных
3. Создание различных графиков и диаграмм

В этой лабораторной работе рассматривается использование библиотеки GD для обработки изображений. GD является внешней библиотекой, доступной в виде модуля PHP. Описано использование функций библиотеки GD для динамического создания изображений средствами PHP и использование HTTP заголовков для указания браузеру, что вывод PHP-скрипта представляет собой изображение.

Примерный ход выполнения работы.

Практическое использование функций библиотеки GD мы рассмотрим на примере скрипта, строящего график биоритмов.

Теория биоритмов гласит, что эмоциональная, физическая и интеллектуальная активность каждого человека циклически изменяются с определенными интервалами. В момент рождения, все три кривые выходят из нулевой точки и изменяются в течение всей жизни. Если кривая проходит выше нулевой отметки - это активная фаза, ниже - пассивная. Момент пересечения нулевой отметки означает критический день: вы подвержены риску эмоциональной, физической или интеллектуальной "катастрофы". А в тот день, когда все три кривые проходят через ноль, лучше вообще воздержаться от работы над важными PHP проектами.

У каждой кривой собственный цикл:

Физическая: 23 дня

Эмоциональная: 28 дней

Интеллектуальная: 33 дня

Кривые представляют собой синусоиды, поэтому для вычисления значений биоритмов мы можем воспользоваться встроенной функцией sin().

Сначала мы должны получить дату рождения и вычислить, сколько дней человек прожил до текущей даты. Результат будет использован для вычисления текущей фазы биоритма.

Порядок действий:

- проверяем, была ли введена дата рождения;
- если нет, отображаем форму для ввода даты;

После проверки правильности ввода вычисляем сколько дней человек прожил до текущей даты используя юлианский календарь.

```
<?php
if(!isset($birthdate))
{
    /* */
}
$daysGone = abs(gregorianToJD($birthMonth, $birthDay,
$birthYear)
 -
 gregorianToJD(date( "m"), date( "d"), date( "Y")));
?>
```

Подготовка к рисованию состоит из вызова нескольких функций GD. Следующий участок кода:

1. Создает изображение нужных размеров.
2. Регистрирует используемые цвета.
3. Очищает изображения, заливая его цветом фона

```
<?php
$image = imageCreate($diagramWidth, $diagramHeight);
$colorBackgr       = imageColorAllocate($image, 192, 1
```

```
92, 192);
$colorForegr       = imageColorAllocate($image, 255, 2
55, 255);
$colorGrid         = imageColorAllocate($image, 0, 0,
0);
$colorCross        = imageColorAllocate($image, 0, 0,
0);
$colorPhysical     = imageColorAllocate($image, 0, 0,
255);
$colorEmotional    = imageColorAllocate($image, 255, 0
, 0);
$colorIntellectual = imageColorAllocate($image, 0, 255
, 0);
imageFilledRectangle($image, 0, 0, $diagramWidth -
 1, $diagramHeight - 1, $colorBackgr);
?>
```

Функция ImageString() рисует строку текста на изображении в заданном месте. Нам необходимо вывести пять строк Первые две будут находиться в верхней части рисунка и отображать дату рождения и текущую дату, остальные три под графиком будут показывать, каким цветом какой биоритм изображен. Использование идентификаторов цветов, записанных в переменные, позволяет ссылаться на цвета, не указывая каждый раз RGB значения.

```
<?php
// рисуем рамку
imageRectangle($image, 0, 0, $diagramWidth -
 1, $diagramHeight - 20,
               $colorGrid);
// рисуем оси
imageLine($image, 0, ($diagramHeight -
 20) / 2, $diagramWidth,
          ($diagramHeight - 20) / 2, $colorCross);
imageLine($image, $diagramWidth / 2, 0, $diagramWidth
/ 2, $diagramHeight - 20,
          $colorCross);
// выводим текст
imageString($image, 3, 10, 10,  "Birthday: $birthDay.$
birthMonth.$birthYear",
```

```
            $colorCross);
imageString($image, 3, 10, 26,  "Today:    ".  date(
"d.m.Y"),  $colorCross);
imageString($image, 3, 10, $diagramHeight -
 42,  "Physical", $colorPhysical);
imageString($image, 3, 10, $diagramHeight -
 58,  "Emotional", $colorEmotional);
imageString($image, 3, 10, $diagramHeight -
 74,  "Intellectual",
            $colorIntellectual);
?>
```

Графики биоритмов представляют собой синусоиды, отличающиеся только периодом, поэтому мы определяем функцию drawRythm() , которая отображает синусоиду с заданным периодом. Эта функция будет вычислять значения биоритмов для каждого дня в заданном периоде и соединять полученные точки линиями.

```
drawRhythm($daysGone, 23, $colorPhysical);
drawRhythm($daysGone, 28, $colorEmotional);
drawRhythm($daysGone, 33, $colorIntellectual);

for($x = 0; $x < $daysToShow; $x++)
    {
        // вычисление фазы синусоиды, соответствующей
определенному дню
        $phase = (($centerDay + $x) % $period) / $peri
od * 2 * pi();
        //  вычисление значения Y
        $y = 1 -
 sin($phase) * (float)$plotScale + (float)$plotCenter;

        // рисуем линию от предыдущей точки до текущей

        if($x > 0)
            imageLine($image, $oldX, $oldY, $x * $diag
ramWidth / $daysToShow,
            $y, $color);
        // сохраняем текущие координаты для использова
ния в следующем проходе цикла
```

```
            $oldX = $x * $diagramWidth / $daysToShow;
            $oldY = $y;
        }
```

Итак, мы нарисовали наш график. Теперь остается отправить это изображение в браузер (рис. 2.1).

```
header("Content-type:  image/png");
// задаем чересстрочный режим
imageInterlace($image, 1);
// делаем цвет фона прозрачным
imageColorTransparent($image, $colorBackgr);
//imageGIF($image);
imagePNG($image);
```

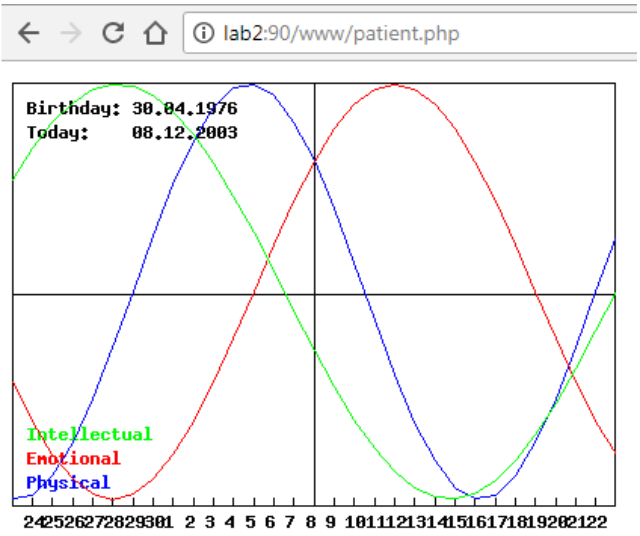

Рис. 2.1. График биоритмов

2.5 Контрольные задания

2.5.1. Варианты заданий для реализации

1. График модели Хольта Винтерса
2. График кубического многочлена вещественной переменной
3. График кубического многочлена целочисленной переменной
4. График параболы
5. График гиперболы

2.5.2. Вопросы для самопроверки

1. Что такое веб-сервер?
2. Примеры веб-серверов.
3. Что такое СУБД?
4. СУБД MySQL, OracleDatabase, PostgreSQL.
5. СУБД MongoDB, Redis.
6. Язык HTML.
7. Язык JavaScript.
8. Система Git.
9. Что такое TDD?
10. Шаги TDD.
11. Основные средства разработки.

3. КРАТКИЙ СИНТАКСИС PHP

3.1. Основные принципы синтаксиса

Язык PHP предназначен для веб-программирования. Синтаксис PHP похож на синтаксис языка C. Программа на PHP обычно находится в файле, который имеет расширение .php. Сам код заключается в «скриптовые» скобки:

<php…>

Код, который содержится внутри этих скобок, выполняется как PHP-инструкции, остальное передается в браузер без изменений. Имена переменных начинаются со знака $. Названия функций завершаются скобками ().

Файл PHP может также содержать HTML-разметку. Для того чтобы закомментировать текст используется знак //. Этот знак применяется для однострочных комментариев. Для комментирования нескольких строк используется многострочный комментарий:

/*…*/

Язык имеет поддержку объектно-ориентированного программирования. В PHP есть возможность создавать классы различных уровней и объекты.

3.2. Работа с данными. Переменные

В PHP существует восемь простых типов данных:

- boolean (логический);
- integer (целочисленный);
- float (числовой с плавающей точкой);
- string (строковый);
- array (массивы);
- object (объекты);
- resource (ресурсы);
- NULL (пустой тип данных).

Типы boolean, integer, float и string относятся к скалярным типам данных. Типы array и object – смешанные типы. Resource и NULL – специальные типы данных.

Тип boolean считается простейшим типом данных, который обозначает истинность значения. Переменные типа boolean могут принимать два значения: TRUE, FALSE.

Тип integer – целые числа, обычно длиной 32 бита. Могут принимать значения от –2 147 483 648 до 2 147 483 647. Числа могут быть указаны в десятичной, восьмеричной и шестнадцатеричной системе счисления. При использовании восьмеричной системы перед числом ставится 0, при использовании шестнадцатеричной – 0x.

Тип float обозначает вещественные числа. Точность числа float хватает для большинства математических вычислений.

String обозначает набор символов любой длины. Строки в PHP могут содержать нулевые символы.

Тип Array – массивы. Массив – упорядоченный набор данных с установленным соответствием между ключом и значением. В простых массивах индексами являются числа, начинающиеся с 0. Если индексом массива является строка, то такой массив называют ассоциативным. Также в PHP есть возможность создания многомерных массивов. Для этого используют функцию array().

Тип object обозначает объекты. Для инициализации объектов используется оператор new. Для доступа к функциям и элементам объекта нужно использовать оператор ->.

Resourse – переменная, содержащая ссылку на внешний ресурс.

«Пустой» тип (NULL) обозначает переменные, не имеющие значений.

В PHP не нужно описывать переменные явно и указывать их тип. Интерпретатор сам определяет тип переменной. Для удаления неиспользуемых переменных используется оператор unset.

Локальные переменные – переменные, объявленные внутри пользовательской функции. На такие переменные можно ссылаться только внутри функции. Значение переменной и она сама удаляются при выходе из пользовательской функции.

Переменные, доступные во всей программе, называются глобальными. Для использования глобальной переменной в пользовательской функции необходимо определить ее внутри функции с помощью ключевого слова global.

Статические переменные существуют только внутри пользовательских функций, однако, их значения сохраняются при выходе из функции. Объявляются статические переменные с помощью ключевого слова static.

Для определения типа переменной используется функция gettype(). Если необходимо изменить тип переменной, нужно использовать функцию settype(). Преобразование типов в PHP происходит в зависимости от контекста.

3.3. Работа с файлами и ресурсами

Обычно работа с файлами состоит из трех этапов: открытие файла, манипуляция с данными и закрытие файла.

Для открытия файла используется функция fopen(). Функция имеет обязательные параметры имя файла и режим открытия. В PHP существуют следующие режимы: r (только для чтения), r+ (для чтения и записи), w (создание нового пустого файла), w+ (если файл существует, открывает его для чтения и записи и удаляет его содержимое), a (открывает файл для записи, указатель сдвигается на конец файла), a+ (открывает файл для чтения и записи, указатель сдвигается на конец файла).

Функция fwrite() записывает данные в файл. В качестве обязательных параметров функция принимает дескриптор и режим файла. Функция fgets() используется для построчного считывания файла. Если файл необходимо читать как единое

целое, нужно использовать функцию readfile(). Для считывания содержимого файла в массив существует функция file(). При необходимости считать файл по символам, можно воспользоваться функцией fgetc().

Для закрытия файла используется функция fclose(). Функция принимает один обязательный параметр.

Преобразование в тип resource не имеет смысла, т.к. он содержит специальные указатели на файлы, области изображения, соединения с БД и т.д. Если на ресурс отсутствуют ссылки, он автоматически освобождается сборщиком мусора.

3.4. Основные конструкции

Конструкции языка PHP похожи на конструкции языка C. Конструкции могут быть сформированы в группу фигурными скобками {}. Основные конструкции языка PHP делятся на условные операторы, циклы, конструкции выбора, конструкции возврата значений, конструкции включений.

Условный оператор if позволяет организовать выполнение кода по условию. Синтаксис конструкции if:

if (выражение) оператор;

Если выражение в скобках после if истинно, то оператор будет выполнен. Иначе оператор исполнен не будет. Если необходимо выполнить несколько операторов, их нужно поместить в фигурные скобки.

Конструкция if-else используется, когда необходимо, чтобы в случае, если условие конструкции if не было выполнено, тоже исполнялись какие-либо операторы.

Синтаксис конструкции if-else:

if (выражение) оператор_1;

else оператор_2;

Еще один вариант синтаксиса конструкции if-else:

if (выражение_1) :

оператор_1;

```
  elseif (выражение_2) :
оператор_2;
  else:
        оператор_3;
  endif;
```

Цикл с предусловием сначала вычисляет значение логического выражения, после чего, если выражение истинно, выполняется тело цикла. Синтаксис конструкции while:

```
while (выражение);
оператор;
```

Цикл с постусловием проверяет значение логического выражения после каждого прохода цикла. Синтаксис конструкции dowhile:

```
do
{
тело_цикла;
}
while (выражение);
```

Если тело цикла необходимо выполнить определенное число раз, используется цикл со счетчиком for. Синтаксис конструкции for:

```
for (инициализирующие_команды; условие; команды_после_итерации)
{
тело_цикла;
}
```

Цикл foreach используется для перебора массивов. Операторы в цикле foreach выполняются для каждого элемента массива. Синтаксис конструкции foreach:

```
foreach (массив as $ключ=>$значение)
операторы;
```

Для того чтобы прервать цикл в ходе итерации, используется оператор break. Необязательный параметр указывает на то, из какого вложенного цикла должен быть

произведен выход. Конструкция continue завершает текущую итерацию и переходит к новой.

Конструкция switch передает управление одному из case-операторов, в зависимости от значения выражения. Синтаксис конструкции switch-case:

```
switch (выражение)
{
case значение_1:
оператор_1;
break;
…
default:  команды_по_умолчанию;
break;
}
```

Оператор return возвращает значения из функций, при этом выполнение функции прерывается.

3.5. Переменные окружения. Куки. Сессии.

Переменные окружения содержат информацию, которой обмениваются браузер и сервер. Их можно использовать как обычные переменные. Переменные окружения можно разделить на следующие группы:

- переменные, которые формируются сервером;
- переменные сервера Apache;
- переменные HTTP-полей запроса;
- переменные SSL-соединения.

Для хранения информации о пользователях при переходах между страницами используются cookies и сессии. Данные cookies хранятся в файлах на компьютере пользователя, а данные сессий хранятся во временных файлах на сервере.

Функция session_start() открывает новую сессию или возобновляет существующую. Для хранения информации

используется массив $_SESSION. Для закрытия сессии используют функцию unset().

Функция setcookie() устанавливает cookie. Для того чтобы вывести cookie испольуют массив $_COOKIE. Чтобы установить срок годности cookies можно использовать функции time() и mktime(). Удаление cookie происходит с помощью функции setcookie().

1. Регулярные выражения – выражения, написанные на специальном языке. Они позволяют осуществлять различные манипуляции с подстроками в тексте, используя метасимволы. Правило для поиска задается строкой-образцом. Если необходимо провести манипуляции с фрагментом текста, дополнительно задается строка замены. Для работы с регулярными выражениями в PHP существуют специальные функции.

Функция intpreg_match проверяет, совпадает ли строка с шаблоном. В зависимости от результата функция возвращает 0 или 1. При указании третьего, необязательного параметра, в него будет занесено первое найденное совпадение.

Функция intpreg_match_all работает аналогично, но возвращает в необязательный параметр все найденные совпадения.

Mix preg_replace позволяет заменять найденные в тексте фрагменты на указанные в параметре.

Для регулярного поиска в массивах используется функция array preg_grep.

2. В языке PHP имеется большое количество математических функций. Среди них можно выделить простые математические функции, такие как: floor() (округляет число в меньшую сторону), ceil() (округляет число в большую сторону), round() (возвращает ближайшее целое число), abs() (модуль числа), min() (наименьший из параметров), max() (наибольший из параметров). Для генерации случайных чисел используются функции rand(), mt_rand(). Для конвертации чисел из одной системы счислений в другую используют

функцию base_convert(). В качестве параметров передается исходное число, основание исходной системы счисления и основание системы счисление результата. Чтобы вычислить квадратный корень числа используется функция sqrt(). Функция log() вычисляет натуральный логарифм числа.

В PHP существуют специальные функции для работы с массивами. Для подсчета элементов массива используется функция count(). Функция print_r() отображает информацию о переменной.

3.6. Лабораторная работа 3. Использование PHP-скриптов в HTML-документах

3.6.1. Постановка задачи

Одним из способов использования скриптов PHP является их непосредственное встраивание в HTML-документы с последующей обработкой их интерпретатором PHP непосредственно перед выдачей их Web-сервером пользователю в виде HTML-страниц. При этом, PHP-скрипт ответственен за формирование только части HTML-документа. Разновидностью этого подхода является использование PHP-скриптов внутри шаблонов html, используемых приложением PHP для компоновки страниц, осуществляемой полностью под управлением PHP-скрипта.

В процессе лабораторной работы необходимо разработать модуль из нескольких страниц с передачей пользовательских данных от одной страницы к другой. При этом, для передачи должны быть использованы следующие механизмы:

1. Механизм передачи переменных в запросе GET
2. Механизм передачи данных через форму POST
3. Механизм передачи данных через COOKIE
4. Механизм передачи данных через сессию

Реализацию модуля необходимо осуществлять в рамках прототипа, разработанного в лабораторной работе №1.

Примерный ход выполнения лабораторной работы.

1. Проведем анализ процесса, который необходимо реализовать, определим его состояния и набор данных, который необходимо транслировать в рамках всего процесса. В нашем случае это будет создание визитки. Определим следующий набор состояний процесса с описанием данных, получаемых в рамках работы с каждым состоянием:

А. Выбор эмблемы визитной карточки. Набор данных:

- url эмблемы

Б. Ввод данных владельца визитки. Набор данных:

- ФИО владельца
- Должность владельца

В. Ввод данных о предприятии. Набор данных:

- Наименование предприятия
- Адрес предприятия

Г. Ввод контактов владельца визитной карточки. Набор данных:

- Email
- Телефон

Д. Вывод сформированной визитной карточки.

2. Создаем заготовки html-страниц Stage1.php, Stage2.php, Stage3.php, Stage4.php и Stage5.php. Формируем в них статический контент и определяем место для динамического контента тегами <?php ?>. В главном меню приложения оформляем ссылку на страницу Stage1.php.

Пример ссылки:

```
      <div class="navbar-header">
      <button type="button" class="navbar-toggle
collapsed" data-toggle="collapse" data-
target="#navbar-main">
          </button>
```

```
          <a class="navbar-brand"
href="Stage1.php">Визитка</a>
        </div>
```

3. На странице Stage1.php выводим список доступных эмблем с возможностью выбора каждой эмблемы путем выбора гиперссылки. Все гиперссылки выбора должны указывать на страницу Stage2.php и содержать в строке параметров ссылки имя выбранной эмблемы. Например:

```
    <ul class="list-group">
      <li class="list-group-item"><a tabindex="-1"
href="Stage2.php?ename=emblem1.php">1-й
вариант</a></li>
      <li class="list-group-item"><a tabindex="-1"
href="Stage2.php?ename=emblem2.php">2-й
    </ul>
```

4. На странице Stage2.php выводим форму ввода данных с типом передачи POST и ссылкой на обработчик формы Stage3.php (рис.3.1):

```
    <form class="well form-inline" name="Stage2Form"
method="post" action="Stage3.php">
      <button type="submit"
class="btn">Отправить</button>
      </form>
```

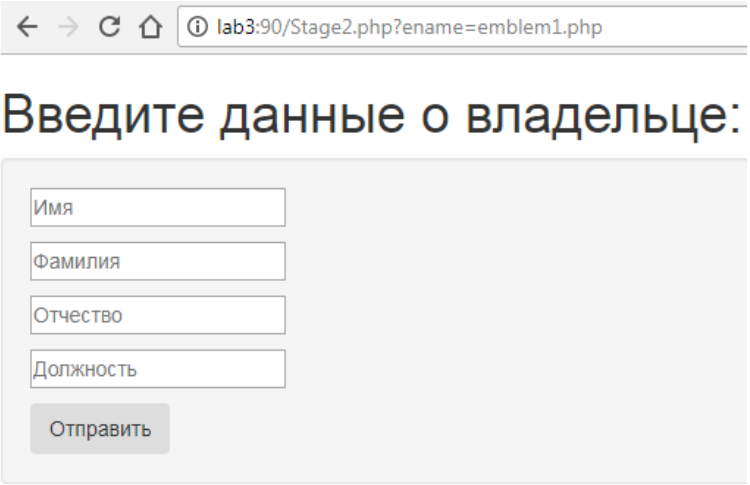

Рис. 3.1. Форма ввода данных

Определяем поля формы, в которые необходимо ввести данные на этом этапе: ФИО владельца, Должность владельца.

```
      <p><input type="text" class="input-small"
name="name" placeholder="Имя"></p>
      <p><input type="text" class="input-small"
name="surname" placeholder="Фамилия"></p>
      <p><input type="text" class="input-small"
name="thirdname"  placeholder="Отчество"></p>
      <p><input type="text" class="input-small"
name="position"  placeholder="Должность"></p>
```

Теперь необходимо выбрать данные, полученные на предыдущем этапе (наименование эмблемы) и переслать их дальше. Выборка осуществляется путем доступа из скрипта php к ассоциативному массиву $_GET. Передача данных посредством формы осуществляется через элемент специального типа hidden:

```
    <input    type=hidden    name=ename    value=<?=
$_GET['ename'] ?> >
```

Обратите внимание на специальный формат вставки php-скрипта, используемый в том случае, когда необходимо вычислить и вернуть какое-либо значение. Дальнейший доступ к данным формы осуществляется при помощи ассоциативного массива $_POST.

5. Ввод данных на страницах Stage3.php и Stage4.php будем также осуществлять при помощи формы. Однако, для данных, введенных ранее, на текущем этапе мы используем механизм файлов COOKIE, позволяющих хранить данные на клиентской стороне. Для этого, обрабатываем массив полученных данных и при помощи команды set_cookie сохраняем переменные в окружении браузера. Не забудьте задать время жизни cookie, по истечении которого они должны стать неактуальными:

```
setcookie("name", $_POST['name']);
setcookie("name", $_POST['name'], time()+3600);
```

6. На странице Stage4.php нам необходимо все данные, полученные в предыдущих этапах, сохранить в сессии. Для этого необходимо при помощи команды session_start инициализировать сессию и передать в нее необходимые значения через ассоциативный массив $_SESSION. При этом, пользователю выдается сгенерированный идентификатор сессии session_id, который передается в дальнейшем через переменные GET/POST, либо сохраняется в браузере в виде cookie. Данные, принадлежащие клиенту, при этом сохраняются на сервере либо в файлах, либо в БД (зависит от настроек), а доступ к этим данным осуществляется при помощи идентификатора сессии. Достанем данные прошлых форм из ассоциативных массивов $_COOKIE и $_POST и сохраним их в качестве переменных сессии:

```
$_SESSION['thirdname'] = $_COOKIE["thirdname"];
```

```
    $_SESSION['position'] = $_COOKIE["position"];
    $_SESSION['name_interprise'] =
$_POST['name_interprise'];
    $_SESSION['adress_interprise'] =
$_POST['adress_interprise'];
```

7. На финальном этапе на странице Stage5.php нам необходимо объединить данные, полученные из сессии, с данными, введенными через форму на этапе Stage4.php и сформировать результат. Это будет выглядеть следующим образом (рис. 3.2):

```
    <form  class="form-horizontal muted" role="form">
      <div class="form-group">
        <label class="col-sm-2 control-
label">Имя:</label>
        <div class="col-sm-10">
          <p class="form-control-static"><?php  echo
$_SESSION['name'] ?></p>
        </div>
      </div>
    <div class="form-group">
        <label class="col-sm-2 control-
label">Фамилия:</label>
        <div class="col-sm-10">
          <p class="form-control-static"><?php  echo
$_SESSION['surname'] ?></p>
        </div>
      </div>
        <div class="form-group">
        <label class="col-sm-2 control-
label">Отчество:</label>
        <div class="col-sm-10">
          <p class="form-control-static"><?php  echo
$_SESSION['thirdname'] ?></p>
        </div>
```

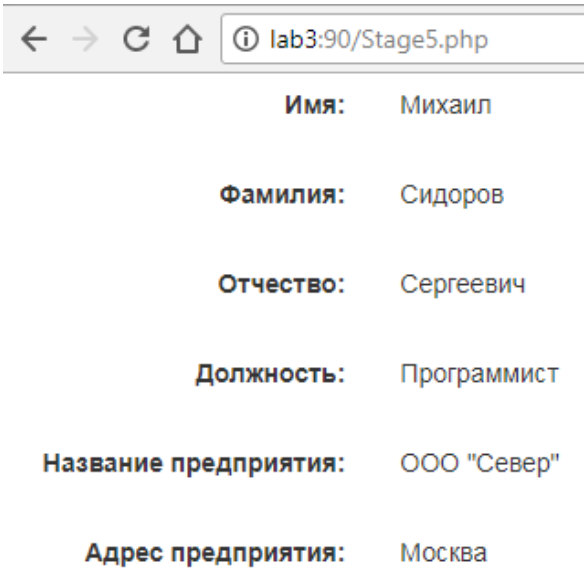

Рисунок 3.2. Визитка

3.7. Контрольные задания

3.7.1. Варианты заданий для реализации

1. Анкета сотрудника
2. Астрологический прогноз на основе года, дня и часа рождения
3. Тест на решение задач (4 задачи)
4. Шуточный тест с разными результатами
5. Оформление доставки товара
6. Формирование комплектации автомобиля
7. Социологический опрос
8. Многоэтапное решение сложной задачи (как с выбором вариантов, так и с вводом ответа)

3.7.2. Вопросы для самопроверки

1. Как в PHP обозначаются переменные?
2. Какие виды комментариев существуют в PHP?
3. Перечислите простые типы данных в PHP.
4. Чем отличаются локальные, глобальные и статические переменные?
5. Какие функции для работы с файлами вы знаете?
6. Синтаксис условного оператора If.
7. Какие циклы существуют в PHP?
8. Конструкция switch-case.
9. Что такое переменные окружения?
10. Для чего используются cookies и сессии?
11. Что такое регулярные выражения?
12. Какие основные функции PHP вы знаете?

4. АРХИТЕКТУРА PHP-ПРИЛОЖЕНИЯ

4.1. ООП в PHP

Класс – это шаблон кода, который используется для создания объектов. Класс определяется с помощью ключевого слова class после которого указывается произвольное имя класса. В имени класса может использоваться любое сочетание букв и цифр, но они не должны начинаться с цифры. Код, связанный с классом должен быть заключен в фигурные скобки, которые указываются после имени. Определение класса описывает, какие элементы будут содержаться в каждом новом экземпляре этого класса.

Так как класс – это шаблон для создания объектов, следовательно, объект – это данные, которые создаются и структурируются в соответствии с шаблоном, определенным в классе. Объект также называют экземпляром класса, тип которого определяется классом. Для создания нового экземпляра класса нам понадобится оператор new. Он используется совместно с именем класса следующим образом:

```
<?php
 // Ключевое слово new сообщает интерпретатору PHP о необходимости
 // создать новый экземпляр класса first
 $obj1 = newfirst();
 $obj2 = newfirst();
?>
```

После оператора new указывается имя класса на основе которого будет создан объект. Оператор new создает экземпляр класса и возвращает ссылку на вновь созданный объект. Эта ссылка сохраняется в переменной соответствующего типа. В результате выполнения этого кода будет создано два объект типа first. Хотя функционально они идентичны (т.е. пусты) $obj1 и $obj2 – это два разных объекта одного типа, созданных с помощью одного класса.

В классе можно определить переменные. Переменные, которые определены в классе называются свойствами (или полями данных). Они определяются с одним из ключевых слов protected, public, или private, характеризующих управление доступом.

Методы – это обычные функции, которые определяются внутри класса, они позволяют объектам выполнять различные задачи. Объявление метода напоминает определение обычной функции, за исключением предваряемого одного из ключевых слов protected, public, или private. Если в определении метода опустить ключевое слово, определяющее видимость, то метод будет объявлен неявно как public. К методам объекта можно обращаться с помощь символов '->', указав объект и имя метода. При вызове метода, так же как и при вызове функции нужно использовать круглые скобки.

У класса может быть определен специальный метод – конструктор, который вызывается каждый раз при создании нового экземпляра класса (объекта) с целью инициализировать его, например установить значения свойств. Конструктор, как и любой другой метод, может иметь параметры. Чтобы определить метод в качестве конструктора его необходимо назвать __construct().

4.2. Шаблоны

GRASP — шаблоны, используемые в объектно-ориентированном проектировании для решения общих задач по назначению обязанностей классам и объектам.

Краткая характеристика девяти шаблонов, из которых состоит GRASP:

1. **Информационный эксперт** (InformationExpert)

Шаблон определяет базовый принцип распределения обязанностей. Обязанность должна быть назначена тому, кто владеет максимумом необходимой информации для исполнения — информационному эксперту.

2. **Создатель** (Creator)

Класс должен создавать экземпляры тех классов, которые он может: содержать или агрегировать, записывать, использовать, инициализировать, имея нужные данные.

3. **Контроллер** (Controller)

Отвечает за операции, запросы на которые приходят от пользователя, и может выполнять сценарии одного или нескольких вариантов использования. Не выполняет работу самостоятельно, а делегирует компетентным исполнителям.

4. **Слабое зацепление** (LowCoupling)

«Степень зацепления» (сопряжения) — мера неотрывности элемента от других элементов (либо мера данных, имеющихся у него о них). «Слабое» зацепление — распределение обязанностей и данных, обеспечивающее взаимную независимость классов.

5. **Высокая степень связности** (HighCohesion)

Предметные области следует разделять по классам.

Связность класса — мера подобия предметных областей его методов:

«Высокая» степень — сфокусированные подсистемы (предметная область определена, управляема и понятна);

«Низкая» степень — абстрактные подсистемы. Затруднены: восприятие, повторное использование, поддержка, устойчивость к внешним изменениям.

6. **Полиморфизм** (Polymorphism)

Устройство и поведение системы определяется данными или задано полиморфными операциями её интерфейса.

7. **Чистая выдумка** (PureFabrication)

Не относится к предметной области, но уменьшает зацепление, повышает связность, упрощает повторное использование. «PureFabrication» отражает концепцию сервисов в модели проблемно-ориентированного проектирования.

8. **Посредник** (Indirection)

Слабое зацепление между элементами системы (и возможность повторного использования) обеспечивается назначением промежуточного объекта их посредником.

9. **Устойчивость к изменениям** (ProtectedVariations)

Шаблон защищает элементы от изменения другими элементами (объектами или подсистемами) с помощью вынесения взаимодействия в фиксированный интерфейс, через который (и только через который) возможно взаимодействие между элементами. Поведение может варьироваться лишь через создание другой реализации интерфейса.

4.3. Принципы SOLID

1. **Принцип единственной ответственности** - обозначает, что каждый объект должен иметь одну ответственность и эта ответственность должна быть полностью инкапсулирована в класс. Все его поведения должны быть направлены исключительно на обеспечение этой ответственности.

2. **Принцип открытости/закрытости** - принцип ООП, устанавливающий следующее положение: «программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения».

3. **Принцип подстановки Барбары Лисков** - объекты в программе могут быть заменены их наследниками без изменения свойств программы.

4. **Принцип разделения интерфейса -** говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться клиенты, которые этот метод не используют.

5. **Принцип инверсии зависимостей -** модули верхних уровней не должны зависеть от модулей нижних уровней. Оба типа модулей должны зависеть от абстракций. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

4.4. MVC

Model-view-controller (**MVC**, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Шаблон MVC описывает простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате, приложение легче масштабируется, тестируется, сопровождается и конечно же реализуется. В архитектуре MVC модель предоставляет данные и правила бизнес-логики, представление отвечает за пользовательский интерфейс, а контроллер обеспечивает взаимодействие между моделью и представлением.

Типичную последовательность работы MVC-приложения можно описать следующим образом:

1. При заходе пользователя на веб-ресурс, скрипт инициализации создает экземпляр приложения и запускает его на выполнение. При этом отображается вид, скажем главной страницы сайта.

2. Приложение получает запрос от пользователя и определяет запрошенные контроллер и действие. В случае главной страницы, выполняется действие по умолчанию (*index*).

3. Приложение создает экземпляр контроллера и запускает метод действия, в котором, к примеру, содержаться вызовы модели, считывающие информацию из базы данных.

4. После этого, действие формирует представление с данными, полученными из модели и выводит результат пользователю.

Модель — содержит бизнес-логику приложения и включает методы выборки (это могут быть методы ORM), обработки (например, правила валидации) и предоставления конкретных данных, что зачастую делает ее очень толстой, что вполне нормально.

Вид — используется для задания внешнего отображения данных, полученных из контроллера и модели.

Контроллер — связующее звено, соединяющее модели, виды и другие компоненты в рабочее приложение. Контроллер отвечает за обработку запросов пользователя. Контроллер не должен содержать SQL-запросов. Их лучше держать в моделях. Контроллер не должен содержать HTML и другой разметки. Её стоит выносить в виды.

4.5. REST-архитектуры

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как WorldWideWeb, который, как правило, используется для построения веб-служб. Системы, поддерживающие REST, называются RESTful-системами.

В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным

идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

Управление информацией сервиса основывается на протоколе передачи данных. Наиболее распространенный протокол конечно же HTTP. Для HTTP действие над данными задается с помощью методов: GET (получить), PUT (добавить, заменить), POST (добавить, изменить, удалить), DELETE (удалить). Таким образом, действия CRUD (Create-Read-Updtae-Delete) могут выполняться как со всеми 4-мя методами, так и только с помощью GET и POST.

Web-сервис – это приложение работающее в WorldWideWeb и доступ к которому предоставляется по HTTP-протоколу, а обмен информации идет с помощью формата XML. Следовательно, формат данных передаваемых в теле запроса будет всегда XML.

Для каждой единицы информации (info) определяется 5 действий. А именно:

GET /info/ (Index) – получает список всех объектов. Как правило, это упрощенный список, т.е. содержащий только поля идентификатора и названия объекта, без остальных данных.

GET /info/{id} (View) – получает полную информацию о объекте.

PUT /info/ или POST /info/ (Create) – создает новый объект. Данные передаются в теле запроса без применения кодирования, даже urlencode. В PHP тело запроса может быть получено таким способом:

```
functiongetBody() {
if (!isset($HTTP_RAW_POST_DATA))
$HTTP_RAW_POST_DATA = file_get_contents("php://input");
return $HTTP_RAW_POST_DATA;
}
```

POST /info/{id} или PUT /info/{id} (Edit) – изменяет данные с идентификатором {id}, возможно заменяет их.

Данные так же передаются в теле запроса, но в отличие от PUT здесь есть некоторый нюанс. Дело в том, что POST-запрос подразумевает наличие urldecoded-post-data. Т.е. если не применять кодирования – это нарушение стандарта.

DELETE /info/{id} (Delete) – удаляет данные с идентификатором {id}.

Архитектура REST очень проста в использовании. По виду пришедшего запроса сразу можно определить, что он делает, не разбираясь в форматах (в отличие от SOAP, XML-RPC). Данные передаются без применения дополнительных слоев, поэтому REST считается менее ресурсоемким.

Самое главное достоинство сервисов в том, что с ними работать может какая угодно система, будь то сайт, flash, программа и др. так как методы парсинга XML и выполнения запросов HTTP присутствуют почти везде.

4.6. ORM

ORM (объектно-реляционное отображение) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Существуют как проприетарные, так и свободные реализации этой технологии.

В объектно-ориентированном программировании объекты в программе представляют объекты из реального мира. В качестве примера можно рассмотреть адресную книгу, которая содержит список людей с нулём или более телефонов и нулём или более адресов. В терминах объектно-ориентированного программирования они будут представляться объектами класса «Человек», которые будут содержать следующий список полей: имя, список (или массив) телефонов и список адресов.

Суть задачи состоит в преобразовании таких объектов в форму, в которой они могут быть сохранены в файлах или

базах данных, и которые легко могут быть извлечены в последующем, с сохранением свойств объектов и отношений между ними. Эти объекты называют «хранимыми» (англ. persistent). Исторически существует несколько подходов к решению этой задачи.

Doctrine — объектно-реляционный проектор (ORM) для PHP 5.3.0+, который базируется на слое абстракции доступа к БД (DBAL). Одной из ключевых возможностей Doctrine является запись запросов к БД на собственном объектно-ориентированном диалекте SQL, называемый DQL (DoctrineQueryLanguage) и базирующийся на идеях HQL (HibernateQueryLanguage).

4.7. Функциональный подход

Функциональное программирование - способ организации вычислений без состояния. Строго говоря, состояние у такой программы конечно есть, это - совокупность контекстов всех её функций.

Типичный функциональный язык программирования поддерживает функции высокого порядка – это функции, которые принимают в качестве аргументов или возвращают другие функции. Большинство из них поддерживает такие вещи, как карринг (currying) и частичное применение функции (partial function application). Также в языках функционального программирования можно встретить тщательно продуманную систему типов, которые используют optiontype для предотвращения появления нулевых указателей, которые стали обычным делом для императивных или объектно-ориентированных языков программирования.

Функциональное программирование обладает несколькими плюсами: отсутствие возни с состояниями делает параллелизм проще, фокусировка на функции — на минимальной единице кода, который можно было бы использовать снова – может привести к интересным вещам,

связанным с их повторным использованием; требование к функциям быть определенными это отличная идея для создания стабильных программ.

PHP не является «настоящим» или «чистым» функциональным языком. Он далек от этого. Тем не менее, в нем есть несколько интересных «строительных блоков» для целей функционального программирования. Для начала, call_user_func, call_user_func_array и $callable(). call_user_func принимает callback-функцию и список аргументов, после чего вызывает этот callback с переданными аргументами. call_user_func_array делает то же самое, за исключением того, что она принимает массив аргументов. Это очень похоже на fn.call() и fn.apply() в JavaScript (без передачи области видимости). Гораздо менее известная, но отличная функция в PHP 5.4 это возможность вызывать функции. callable это мета-тип в PHP (то есть состоящий из нескольких вложенных типов): callable может быть строкой для вызова простых функций, массивом из <string,string> для вызова статичных методов и массивом из <object,string> для вызова методов объекта, экземпляра Closure или чего угодно, осуществляющего метод __invoke().

В PHP 5.4 появился новый тип “callable”, который позволяет простой доступ к мета-типу callable.

4.8. Лабораторная работа №4. Разработка ООП-приложения PHP

4.8.1. Постановка задачи

Разработка современных PHP-приложений немыслима без использования объектно-ориентированного подхода. При этом, основной целью использования ООП в разработке PHP приложений в подавляющем большинстве случаев является необходимость декомпозиции. Это связано со спецификой конструирования PHP-приложений, точкой старта которых

является поступление на Web-сервер HTTP-запроса, а финалом – формирование HTTP-ответа. Схема MVC в качестве исходного шаблона вполне подходит для проектирования и реализации PHP-приложений.

Вместе с тем, даже в рамках этой схемы использование таких принципов ООП, как наследование и полиморфизм, а также шаблонов GRASP, позволяет добиться необходимой гибкости приложения. Рассмотрим одну из задач, в которой использование объектно-ориентированного подхода, позволяет существенно увеличить гибкость приложения и обеспечить его легкую модификацию.

Рассмотрим Internet-приложение, состоящее из нескольких блоков функциональности, причем экранная форма пользователя может настраиваться, исходя из необходимости использования блоков, В общем виде, главная форма выглядит так (рис. 4.1):

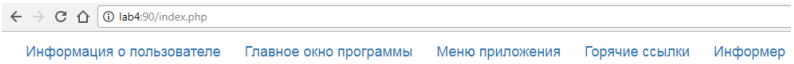

Рис. 4.1. Главная форма

И состоит из следующих компонентов:

1. Информация о пользователе
2. Главное окно программы
3. Меню приложения
4. Горячие ссылки
5. Информер

При этом, в зависимости от пользователя содержание и состав блоков может меняться. Для организации блочной компоновки создадим иерархию объектов с общим предком VisualBlock, который будет иметь виртуальную функцию Draw. Эту функцию во всех предках объектов необходимо будет перегрузить. Код базового класса:

```
class VisualBlock {

    static function Draw() {
       echo "<h1>You must to overload this
method</h1>";
    }
}
```

Далее, от базового класса VisualBlock наследуем классы, отображающие следующие компоненты:

UserInfoBlock – информация о пользователе

MainMenuBlock – главное меню программы

SecondMenuBlock – меню приложения

HotLinksBlock – блок горячих ссылок

CustomInfoBlock – блок информера

В каждом из этих классов мы перегружаем виртуальную функцию Draw, реализуя отображение нужного типа блоков:

```
class UserInfoBlock extends VisualBlock {
      static public function Draw() {
           echo "<h1>This is UserInfoBlock</h1>";
       }
   }
     class MainMenuBlock  extends VisualBlock {
     static public function Draw() {
           echo "<h1>This is MainMenuBlock
</h1>";
       }
    }
```

Создаем файл, реализующий класс UserConfig и реализуем в нем статичный метод get, возвращающий конфигурацию экрана для текущего пользователя. Метод должен возвращать массив классов. Код класса UserConfig:

```
class UserConfig {
    static function get() {
        $vb[0] = UserInfoBlock:: Draw();
```

```
            $vb[1] = MainMenuBlock:: Draw();
            return $vb;
        }
    }
```

При этом, инициализируем этот массив нужными нам визуальными блоками:

```
$vb[0] = UserInfoBlock:: Draw();
$vb[1] = MainMenuBlock:: Draw();
```

Формируем страницу вывода информации на экран. При этом, php-код на данной странице будет являться внедренным и выполнять функции отрисовки блоков внутри html-документа. В месте, где будут расположены блоки, вставляем следующий код (рис. 4.2):

```
<div>
class UserConfig {
    static function get() {
        $vb[0] = UserInfoBlock::Draw();
        $vb[1] = MainMenuBlock:: Draw();
        return $vb;
    }
} </div>
```

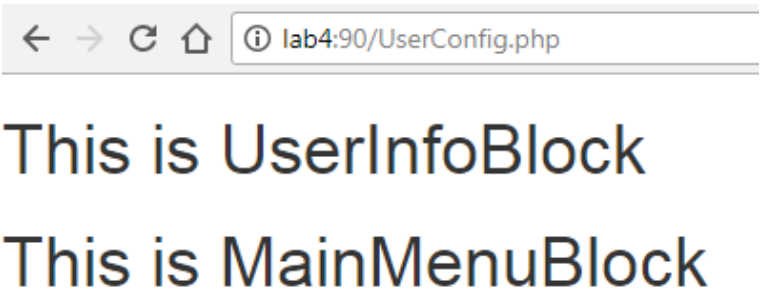

Рис. 4.2. Внедренный код

4.9. Контрольные задания

4.9.1. Варианты заданий для реализации

1. Рабочее место диспетчера автоперевозок
2. Рабочее место трейдера Forex
3. Рабочее место политолога
4. Рабочее место механика авиалайнера
5. Страница управления «умным домом»
6. Страница контроля серверной нагрузки
7. Рабочее место технолога цеха
8. Рабочее место маркетолога

4.9.2. Вопросы для самопроверки

1. Что такое класс и объект?
2. Как в PHP определяется класс?
3. Как создать объект в PHP?
4. Что такое методы?
5. Перечислите шаблоны, из которых состоит GRASP?
6. Дайте определение MVC.
7. Опишите последовательность работы MVC.
8. Что такое модель, вид, контроллер?
9. Что такое REST?
10. Технология ORM.
11. Что такое функциональное программирование?

5. ТЕХНИЧЕСКИЕ ОСОБЕННОСТИ РАБОТЫ

5.1. Работа с сетью

Заголовки HTTP используются для "общения" браузера и web-сервера, например, когда браузер запрашивает какой-либо документ, он посылает заголовок GET, а когда сервер возвращает тип документа, то он делает это ни как-нибудь, а в заголовке Content-type.

Ниже приведен список и краткое описание основных заголовков HTTP.

Заголовок Accept предназначен для информирования сервера о типах данных, которые поддерживаются клиентом (браузером). В этом заголовке браузер перечисляет, какие типы документов он "понимает". Перечисление идет через запятую. Используется переменная окружения HTTP_ACCEPT.

Заголовок Content-type предназначен для идентификации типа передаваемых данных. При этом заголовок Content-type использует переменную окружения CONTENT_TYPE. Обычно для этого заголовка указывается значение application/x-www-form-urlencoded. Таким образом, указывается формат, в котором все управляющие символы (т.е. символы, не являющиеся алфавитно-цифровыми) специально кодируются. Это тот самый формат передачи, который используется методами GET и POST.

Заголовок Content-length содержит строку, в которой записана длина передаваемых данных в байтах при использовании метода передачи POST. За заголовком закреплена одноименная переменная CONTENT_LENGTH. Если задействуется метод GET, то этот заголовок отсутствует, и значит, переменная окружения не устанавливается.

Заголовок Cookie хранит все Cookies. Данный заголовок использует переменную окружения HTTP_COOKIE. Для установки Cookies используется заголовок Set-Cookie.

Заголовок GET использует следующие переменные окружения:

REQUEST_URI - запрашиваемый идентификатор ресурса;

QUERY_STRING - передаваемые сценарию параметры;

REQUEST_METHOD - метод передачи информации. В данном случае эта переменная будет содержать значение GET.

Получив заголовок Location вместе с указанным в нем URL, сервер немедленно переходит по указанному URL, не дожидаясь, пока тело документа загрузится.

Заголовок POST использует те же переменные окружения, что и заголовок GET (переменная REQUEST_METHOD содержит значение POST). Напомним, что данные методом POST можно передавать в конце заголовков.

Заголовок Pragma используется для различных целей, одна из которых - это запрет кэширования документа. Пример заголовка: Pragma: no-cache

Заголовок Server содержит название и версию программного обеспечения сервера.

С помощью заголовка Referer можно узнать ссылающийся на нас сервер. Это бывает полезно, когда нам необходимо узать обратные ссылки, например, при анализе эффективности сетевой рекламы. Переменная окружения: HTTP_REFERER.

Заголовок User-Agent содержит версию браузера. Переменная окружения: HTTP_REFERER.

В PHP есть встроенные функции для работы с заголовками HTTP. Для передачи заголовков HTTP предназначена функция header().

Сокет - это интерфейс взаимодействия клиента и сервера. Интерфейс - это, в свою очередь, правило, по которым происходит общение между клиентом и сервером.

Создание сокета, готового для подключения к другому сокету в Internet, выполняется с помощью функции socket_connect (): socket_connect($socket, $address [, $port]).

При выполнении эта функция подключается к указанному серверу, используя предоставленный сокет, и возвращает булевское значение, указывающее на то, успешно ли выполнен запрос.

После установки соединения с другим прослушивающим сокетом/сервером - данные могут быть отправлены и приняты через сокет с помощью функций socket_read() и socket_write(): socket_write($socket, $buffer [, $length]); socket_read($socket, $length [, $type]).

5.2. Обработка форм. Валидация данных

Одним из основных способов передачи данных веб-сайту является обработка форм. Формы представляют специальные элементы разметки HTML, которые содержат в себе различные элементы ввода - текстовые поля, кнопки и т.д. И с помощью данных форм мы можем ввести некоторые данные и отправить их на сервер. А сервер уже обрабатывает эти данные.

Создание форм состоит из следующих аспектов:

- Создание элемента <form><form> в разметке HTML
- Добавление в этот элемент одно или несколько поле ввода
- Установка метода передачи данных: GET или POST
- Установка адреса, на который будут отправляться введенные данные

Все данные, которые вы хотите получить из HTML-формы в PHP сценарий, обрабатываются с помощью суперглобальных массивов $_POST или $_GET, в зависимости от указанного в атрибуте method метода передачи данных.

Проверке корректности данных, вводимых пользователем необходимо уделять достаточно большое внимание, поскольку необработанные ошибки, возникающие

при вводе неправильном вводе данных, приводят к ошибкам в работе скрипта, зачастую катастрофическим.

Кроме чистых ошибок пользователя, необходимо также исключить ситуации, в которых возможно злонамеренное введение некорректных данных, к примеру, различных скриптов. Для этого вводимый пользователем текст необходимо обработать функциями удаления HTML-тегов (для исключения возможности написания скриптов на JavaScript и Visual Basic) и обратных слешей (для исключения возможности написания скриптов на Perl). Т. о. минимальный набор действий, необходимый для проверки корректности данных, вводимых пользователем, включает следующие этапы:

- проверка того, что пользователь ввел данные;
- проверка допустимости вводимых пользователем данных (как правило, осуществляется при помощи регулярных выражений);
- обработка текста, введенного пользователем функцией htmlspecialchars для удаления HTML-тегов;
- обработка текста, введенного пользователем функцией stripslashes для удаления обратных слешей.

5.3. Работа с БД

В PHP реализована обширная поддержка практически всех существующих серверов баз данных.

Соединение с сервером БД осуществляется при помощи функции mysql_connect: $connect = mysql_connect(<хост>, <логин>, <пароль>); По умолчанию, на mysql-сервере в таблице пользователей есть пользователь root, который может иметь доступ только с localhost, то есть с того же самого компьютера, где стоит сервер mysql.

Механизм работы функций запросов к БД такой же, как и у функции соединения: функции передаются параметры

запроса и (если надо) соединения, а результат записывается в переменную:

```
    $result = mysql_db_query(string база данных,
string запрос [, переменная соединения]);
    или
    $result = mysql_query(string запрос [, переменная
соединения]);
```

Чтобы использовать функцию mysql_query, в которой база данных не указывается, надо предварительно выбрать используемую базу данных:

```
    mysql_select_db(string базаданных);
```

Теперь у нас есть переменная $result. Это указатель на результат выполнения запроса. Получить строки можно через функции mysql_fetch_row и mysql_fetch_array:

```
    echo "<table>";
    while ($row = mysql_fetch_array($result))
      echo "<tr><td>", $row["field1"], "</td><td>",
$row["field2"], "</td></tr>";
    echo "</table>";
```

Функция mysql_fetch_array выдаёт в указанную переменную (в данном случае $row) массив, индексы которого - имена полей. mysql_fetch_row выдаёт массив, индексы которого - числа, начиная с 0.

Запросы-действия - это команды DELETE и UPDATE. Подобные запросы - в "правах" такие же, как и SELECT, поэтому отправка команды серверу происходит тем же способом - mysql_query (mysql_db_query). Но в данном случае функция не возвращает результата.

Сообщение о последней ошибке можно получить через функцию mysql_error:

```
    echo "Ошибка базы данных:", mysql_error();
```

5.4. Безопасность

Безопасность - один из ключевых моментов, на которые должен в первую очередь обращать внимание любой разработчик. Взлом сайтов - это абсолютно незаконное, но иногда очень прибыльное занятие, а лёгкие деньги привлекают как начинающих, так и опытных "специалистов". Защиту сайта необходимо вести в двух направлениях: защита кода и защита данных.

Существует множество вариантов атак на сайты, но львиная доля взломов происходит из-за халатности администраторов и программистов. Все наиболее популярные атаки можно разделить на три группы: атаки на сервер, атаки на интерпретатор и атаки на скрипт.

При атаках на сервер злоумышленники используют уязвимости в серверных программах (например, уязвимости веб-сервера или почтовых служб). Единственное средство защиты от таких атак - качественная работа администратор сервера, отслеживание ими новостей о новых уязвимостях и своевременное обновление программного обеспечения.

Второй тип - атаки на интерпретатор, а в нашем случае - на интерпретатор PHP. Не секрет, что в любой программе есть "дырки", которыми может воспользоваться злоумышленник. Смысл атаки в том, чтобы подать на вход интерпретатору такую последовательность команд, исполнив которую он ошибочно предоставит злоумышленнику доступ к каким-либо ресурсам сервера. Защита от таких атак - своевременное обновление администраторами версии PHP и установка пакетов исправлений. Со своей стороны разработчик должен отслеживать появляющиеся уязвимости в функциях и соответствующим образом изменять скрипты.

Защита от третьего вида атак полностью лежит на плечах программиста. Как уже говорилось, большая часть взломов происходит из-за ошибок или недоработок в коде. Чтобы

избежать основных ошибок, следуйте следующим простым правилам:

1. Никогда не доверяйте чужим данным, даже если они получены из надёжного, но всё-таки стороннего источника (например, с сайта партнёра или от зарегистрированного пользователя). Все данные должны быть в обязательном порядке перед использованием проверены на корректность.

2. Пользуйтесь готовыми библиотеками доступа к базам данных. Такие библиотеки обычно хорошо проверены на безопасность и постоянно совершенствуются хорошими специалистами. Вам достаточно лишь следить за обновлениями и поддерживать свои версии библиотек в актуальном состоянии.

3. Не доверяйте пользователям даже в мелочах. Исходите из постулата, что если есть возможность навредить, то найдётся человек, который навредит.

4. Старайтесь самостоятельно обрабатывать ошибки и никогда не показывайте подробности ошибок пользователям. Знание путей к файлам, имён библиотек, названий баз данных и таблиц сильно упростит задачу хакера.

5. Всегда старайтесь очищать используемые переменные. Одна не очищенная вовремя переменная может перечеркнуть все ваши усилия по защите. Это относится и к переменным классов. Если переменная используется только в одной функции - не надо делать её глобальной или представлять как отдельное свойство.

6. Старайтесь не хранить приватные данные на сервере, а если хранить всё-таки приходится - делайте это в базе данных в зашифрованном виде. Это сильно усложнит работу хакера, даже если он уже забрался на ваш сайт.

Необходимость хранения приватных данных на сайте (например, базы клиентов), вынуждает разработчиков встраивать в сайт специальные системы защиты. Чаще всего используются два варианта: HTTP-аутентификация или парольная защита.

HTTP-аутентификация более надёжна, но менее удобна в администрировании. Кроме того, пользователь обычно не может самостоятельно зарегистрироваться - его должен зарегистрировать один из администраторов или модераторов сайта.

Парольная защита требует от программиста больше усилий по разработке и отладке, является менее надёжной, но при этом даёт больше возможностей пользователям по управлению своим аккаунтом.

На сайтах зачастую используются оба метода одновременно: парольную защиту устанавливают для обычных пользователей и HTTP-аутентификацию для входа в раздел администраторов.

5.5. Прием файлов

Загрузка файлов на сервер осуществляется пользователями сети интернет довольно часто, а именно:

- Веб-интерфейсы почтовых сервисов, которые позволяют добавлять к письму приложение (attach), а для этого нужно сначала загрузить файл на сервер, и только после этого его можно добавлять к письму;
- Интерактивные фотогалереи и фотоальбомы, которые не могут существовать без механизма загрузки файлов на сервер;
- Порталы бесплатного программного обеспечения, которые используют для обмена файлами различных программ, и.т.д.

Загрузка файла на сервер осуществляется с помощью multipart-формы, в которой есть поле загрузки файла. В качестве параметра enctype указывается значение multipart/form-data:

```
<form action=upload.php method=post
enctype=multipart/form-data>
```

```
<input type=file name=uploadfile>
<input type=submit value=Загрузить></form>
```

Multipart-формы обычно используют метод передачи POST. Как видно из предыдущего примера, данная форма имеет два поля:

Поле выбора файла для закачки <INPUT type=File>;

Поле указания имени файла, которое он должен будет иметь на сервере <INPUT type=text>.

5.6. Работа с изображениями

PHP можно использовать для создания и манипулирования изображениями различных форматов, среди которых gif, png, jpg, wbmp, andxpm. Кроме того PHP способен выводить поток изображения прямиком в браузер. Для работы вам понадобится PHP собранный с графической библиотекой GD . GD и PHP могут зависеть и от других библиотек, в зависимости от того, с какими форматами изображений вы будете работать.

При помощи расширения EXIF вы сможете обрабатывать информацию хранящуюся в заголовках JPEG и TIFF изображений. При помощи него вы сможете получить доступ к мета тегам генерируемым цифровыми камерами. Для работы EXIF расширения библиотека GD не требуется.

resource imagecreatetruecolor (int $width , int $height) – функция принимает два целочисленных аргумента. Длина и ширина нового изображения. Результат функция возвращает ресурсного типа данных.

imagecolorallocate – функция генерирует представления цвета в том виде, в который используется в картинке, для дальней работы с этим цветом на картинке. Самый первый вызов этой функции задаёт фон изображения.

int imagecolorallocate (resource $image , int $red , int $green , int $blue) –

три параметра – числа от 0 до 255 выражающие цвет в системе RGB.
Первый же параметр функции resource $image это идентификатор, который получили в imagecreatetruecolor. Параметр этот требуется для того, чтобы представление цвета соответствовало требованиями формата изображения.

bool imageline (resource $image , int $x1 , int $y1 , int $x2 , int $y2 , int $color) – функция рисует линию на изображении.

bool imagecopyresampled (resource $dst_image , resource $src_image , int $dst_x , int $dst_y , int $src_x , int $src_y , int $dst_w , int $dst_h , int $src_w , int $src_h) – функция вырезает прямоугольный кусочек из одной картинки, изменяет её размер, сжимает или расширяет, растягивает или сужает и вставляет в новую картинку.

imagecreatefromgif – функция выдает указатель на картинку в памяти с которым можно работать.

imagedestroy – переменную $img указатель она не удаляет, но память на которую переменная указывает она очищает.

5.7. E-mail

Очень часто на сайтах возникает необходимость отправить письмо администартору или клиенту. Сделать это можно двумя способами: на стороне клиента и на стороне сервера.

Для отправки почты со стороны клиента достаточно в HTML-код страницы добавить ссылку"mailto:" с указанием необходимого адреса:

```
<a href="mailto:webmaster@server.com">Написать вебмастеру</a>
```

Когда пользователь кликает по ссылке, браузер запускает почтовый клиент по-умолчанию и помещает указанный адрес в поле "Кому", так что пользователю достаточно написать

текст и отправить письмо. Этот способ никак не связан с PHP, но пользоваться им иногда стоит.

В PHP предусмотрена функция - mail(). Способов использования этой функции несколько. Прототип функции mail() выглядит следующим образом:

```
boolmail(stringto,
      string subject,
       string message
      [, string additional_headers
      [, stringadditional_parameters]])
```

При отправке почты необходимо указать как минимум три параметра: получателя, тему и текст сообщения.

В качестве адреса получателя "to" можно указывать как отдельный адрес, так и целый список, разделённый запятыми.

Кроме обязательных элементов можно указывать два дополнительных - расширенные заголовки и расширенные параметры. Расширенные заголовки служат для добавления к письму дополнительных атрибутов и блоков, таких как адрес отправителя, адрес отправки копии ("cc"), адрес отправки скрытой копии ("bcc") и т.д. Расширенные заголовки так же позволяют отправлять сообщения в HTML-формате.

Расширенные данные - это блок данных, предназначенный для программы, которая будет отправлять почту. Эти данные в некоторых случаях позволяют осуществить тонкую настройку программы для конкретных нужд.

Запустить отправку писем можно двумя способами - в результате действий пользователя (например, после обработки данных формы), либо посредством запланированного задания. Первый случай очевиден - пользователь отсылает форму на сервер, там происходит обработка данных и на указанный почтовый ящик отправляются результаты обработки формы (подтверждение регистрации, счета к оплате, содержимое заказа или что-то еще).

Второй способ чаще всего используется, когда скрипт необходимо выполнять периодически (например, ежедневная рассылка новостей). Организация планирования выполнения задачи средствами PHP - задача нетривиальная. Более простой путь - воспользоваться готовыми инструментами.

Для работы с электронной почтой уже создано множество классов, поэтому вам нет острой нужды писать весь код самостоятельно. Достаточно посетить соответствующий сайт и скачать одну из бесплатных библиотек.

5.8. JSON, AJAX

JSON (JavaScriptObjectNotation) – это простой, основанный на тексте, способ сохранять и передавать структурированные данные. С помощью простого синтаксиса вы можете легко сохранять как простые цифры и строки, так и массивы, объекты, используя при этом не что иное как текст. Так же можно связывать объекты и массивы, что позволяет создавать сложные структуры данных.

После создания JSON строки, ее легко можно пересылать в любое приложение или компьютер, так как это всего лишь текст.

JSON имеет много преимуществ:

1. Он компактный
2. Он понятен для людей и легко считывается компьютером
3. Его легко можно преобразовать в программные форматы: числовые значения, строки, булевой формат, нулевое значение, массивы и ассоциативные массивы.
4. Почти все программные языки имеют функции, позволяющие считывать и создавать json формат данных.

В основном JSON используется для обмена данными между javascript и серверной стороной (php). Другими

словами, для технологии ajax. Это очень удобно, когда вы передаете несколько переменных или целые массивы данных.

Ниже предоставлены основные правила создания JSON строк:

- JSON строка содержит как массив значений, так и объект (ассоциативный массив с парами имя/значение).
- Массив должен быть обвернут в квадратные скобки, [и], может содержать список значений, которые отделяются через кому.
- Объекты обворачиваются с помощью фигурных дужек, { и }, также содержат разделенные комой пары имя/значение.
- Пары имя/значение состоят изимя поля (в двойных кавычках), после чего следует двоиточие (:) , после значение данного поля.
- Значения в массиве или объекте могут быть:

– Числовые (целые или дробные с точкой)
– Строковые (обвернутые в двойные кавычки)
– Логические (true или false)
– Другие массивы (обвернутые в квадратные скобки [и])
– Другие объекты (обвернутые в фигурные дужки { и })
– Нулевое значение (null)

AJAX - это асинхронный (т.е. браузер, отослав запрос, может делать что угодно, например, показать сообщение об ожидании ответа, прокручивать страницу, и т.п.) JavaScript и XML. Он используется для создания динамических и быстрых веб-страниц. AJAX позволяет нам обновлять часть веб-страницы без перезагрузки страницы целиком.

Применение ajax+ php может быть разнообразным, единственное, то что, нельзя конструировать элементы страницы с помощью данной технологии, которые несут в себе релевантность для поисковых систем. Потому что ajax подгружает элементы страницы после ее загрузки при вызове js событий, но как нам известно, поисковые системы не умеют читать javascript кода, поэтому нужно тщательно выбирать где нужно, а где не нужно применять ajax с php.

По сути, технология ajax не может существовать без php скриптов, так как ajax, только отправляет данные на сервер и принимает обратный ответ, при этом не перезагружая страницы. Поэтому правильнее задать вопрос, как связать работу ajax и php. Но об этом мы поговорим в следующем пункте статьи (ajaxphp пример), а сейчас разберемся с спецификой работы ajax.

Для отправки данных на сервер, нужно создать объект XMLHTTPRequest. С помощью него открыть url (php скрипт), послать на него данные (POST или GET метод), получить ответ, и средствами знаний языка js вывести полученный ответ сервера на монитор (ответом может быть любой фрагмент или элемент страницы сайта).

5.9. XML

Язык Extensible Markup Language (XML) можно назвать и языком разметки, и форматом хранения текстовых данных. Это подмножество языка Standard Generalized Markup Language (SGML); он предоставляет текстовые средства для описания древовидных структур и их применения к информации. XML служит основой для целого ряда языков и форматов, таких как Really Simple Syndication (RSS), Mozilla XML UserInterfaceLanguage (XUL), Macromedia Maximume Xperience Markup Language (MXML), Microsofte Xtensible Application Markup Language (XAML) и opensource-язык Java XML UI Markup Language (XAMJ).

Базовым блоком данных в XML является элемент. Элементы выделяются начальным тегом, таким как <book>, и конечным тегом, таким как </book>. Каждому начальному тегу должен соответствовать конечный тег. Если для какого-то начального тега отсутствует конечный тег, XML-документ оформлен неправильно, и синтаксический анализатор (парсер) не сможет проанализировать его надлежащим образом. Названия тегов обычно отражают тип элемента. Можно

ожидать, что элемент book содержит название книги. Текст, содержащийся между тегами, включая пробелы, называется символьными данными.

Имена XML-элементов и атрибутов могут состоять из латинских букв верхнего (A-Z) и нижнего (a-z) регистров, цифр (0-9), некоторых специальных и неанглийских символов, а также трех знаков пунктуации: дефиса, знака подчеркивания и точки. Другие символы в именах не допускаются.

Каждый документ XML содержит один и только один корневой элемент. Корневой элемент — это единственный элемент XML-документа, для которого нет родительского элемента.

Кроме вложенных элементов, что создает отношения родительский-дочерний, XML-элементы могут иметь атрибуты. Это пары имя-значение, присоединенные к начальному тегу элемента. Имена отделяются от значений знаком равенства, =. Значения заключаются в одинарные или двойные кавычки. XML-разработчики практикуют разные подходы к использованию атрибутов. Большую часть информации, содержащейся в атрибуте, можно поместить в дочерний элемент. Некоторые разработчики настаивают на том, чтобы информация атрибутов состояла не из данных, а из метаданных, то есть сведений о данных. Сами данные должны содержаться в элементах. На самом деле решение о том, использовать ли атрибуты, зависит от природы данных и от того, как они извлекаются из XML.

5.10. Лабораторное задание №5. Работа с БД

Лабораторная работа посвящена изучению функций работы с реляционной БД. Наиболее традиционным решением для приложений, выстраиваемых на базе PHP, является использование сервера MySQL. PHP имеет внутренние средства поддержки работы с этой БД, в настоящее время наиболее популярна библиотека mysqli.

Рассмотрим процесс построения приложения, использующего реляционную БД. Для примера возьмем приложение, поддерживающее работу кадровых служб и позволяющее осуществлять учет сотрудников предприятия, а также контролировать использование выданных им подотчетных денег. Первым этапом разработки такого приложения будет являться проектирование реляционной БД. Модель БД будет иметь вид, как показано на рис. 5.1, и состоять из следующих таблиц:

S_dept – подразделения предприятия,

S_emp – сотрудники предприятия,

S_money – денежные транзакции сотрудников предприятия

S_docs – документы, подтверждающие траты сотрудников.

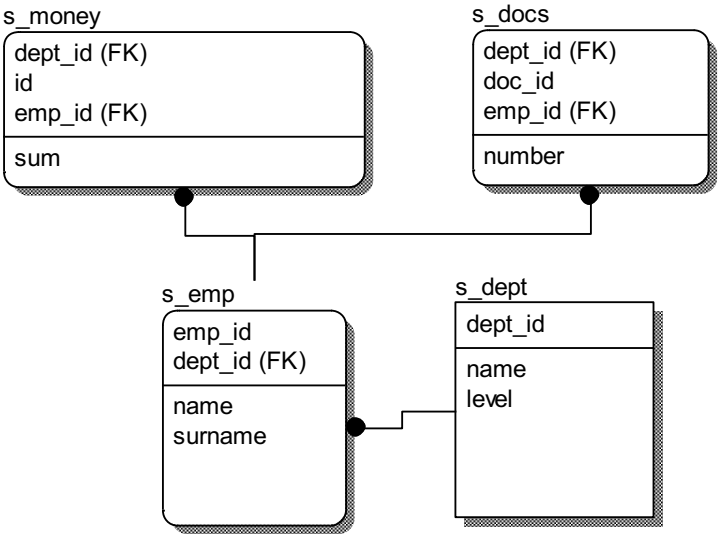

Рисунок 5.1. Модель БД

Создадим эти таблицы и заполним таблицу s_dept несколькими строками. Далее, реализуем функциональность, обеспечивающую ведение данных в таблице s_emp. Для начала, реализуем модуль, выводящий общий список данных. Подключение к таблице будет осуществляться с помощью следующих вызовов:

```
     $link = mysql_connect('localhost',
'mysql','mysql','frames');
     if (!$link)
     {
     echo "<h2>MySQL Error!</h2>";
     }
     $db_selected = mysql_select_db('mysql',$link);
     if (!$db_selected) {
       die ('Не удалось выбрать базу frames:'.
mysql_error());
     }
```

При этом, такие параметры, как $db_user, $db_passw, $db_base, необходимо инициализировать в файле config.php.

Реализуем вывод на экран списка отделов. Для этого сформируем необходимый запрос:

```
     $query = mysql_query ("select * from s_dept order
by name");
     $num_fields = mysql_num_fields($query);
```

И далее выведем его в процессе изображения на экране всей страницы. Вывод можно осуществлять в табличной форме с помощью оператора printf. Оператор может выводить информацию, аналогично оператору echo, но позволяет параметризовать выводимые строки. Смешанное использование в процессе вывода операторов echo и printf не рекомендуется разработчиками php (рис. 5.2).

```
     print "<table border=\"1\" width=\"70%\"
bgcolor=\"WhiteSmoke\">";
```

```
     print "<tr>";
     for ($i=0; $i<$num_fields; $i++)
     {
     print "<th>".mysql_field_name($query,$i)."</th>";
     }
     print "<th></th><th></th>";
     print "</tr>";
     for ($i=0; $i<mysql_num_rows($query); $i++)
     {
     print "<tr>";
     $arr = mysql_fetch_array($query);
     for ($j=0; $j<$num_fields; $j++) {
      if ($j == 0) {$id = $arr[$j]; }
     print "<td>$arr[$j]</td>";
     }
     print "<td><a href=editdept.php?id=$id><img
src=edit.jpg></a></td>";
     print "<td><a href=removedept.php?id=$id><img
src=remove.jpg></a></a></td>";
     print "</tr>";
     }
     print "</table>";
```

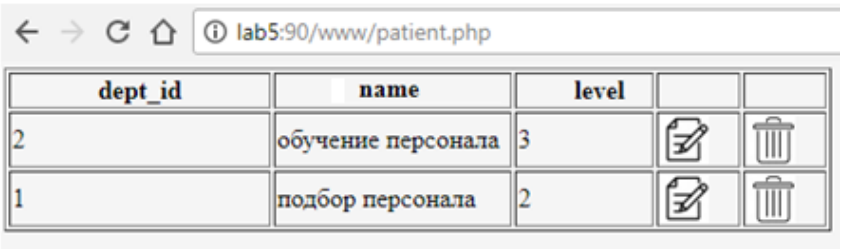

Рисунок 5.2. Вывод результата запроса в html-страницу

Редактирование и удаление строк реализуется здесь с помощью ссылок на модули editdept и removedept с указанием параметра id строки.

К выводимому списку отделов добавим форму, с помощью которой можно вводить в таблицу s_emp новые строки.

```
    printf("<form                                method=post
action=applydept.php>");
    printf(   "<input   type   =hidden   name   =   id
value=''>");
    printf( "<input type=text name=name value=''>");
    printf("<input       type=text       name=surname
value=''>");
    printf("<input type=text name=desp value=''>");
    printf("<input type=submit value=Apply>");
    printf("</form>");
```

Обработчиком данных формы мы назначаем модуль applydept.php, переход по ссылке к нему осуществляется при нажатии кнопки Apply. В файле applydept.php мы осуществляем обработку данных, пришедших из формы. В частности, здесь мы их заносим в базу данных:

```
    if ($_POST['id'] == "")
    {
        $name = $_POST['name'];
        $surname = $_POST['surname'];
        $desp = $_POST['desp'];
        $q      =      "INSERT      INTO      s_emp
(emp_id,name,surname,desp)
    VALUES('','$name','$surname','$desp')";
    $result = mysql_query($q) or die(mysql_error());
    }
```

Приведенный здесь обработчик вносит строку в базу данных при условии, что пришедший параметр id (не показываемый, но содержащийся в форме) имеет пустое значение. Код можно дополнить вариантом, что в случае отличия параметра id от пустого значения, данные нужно изменить в уже существующей строке. После этого, приведенный листинг можно использовать и для создания формы редактирования данных в том случае, если заранее установить параметр id.

Также, hidden элементы часто используются для передачи параметров от формы к форме.

В последнее время наиболее распространенным является вариант, когда для вывода формы на экран и обработки данных, в нее полученных, используется один и тот же модуль (или даже процедура). Это удобно в том случае, когда необходимо осуществлять валидацию данных и в случае проблем с ними, выводить ту же самую форму с частично введенными данными и подсказками, какие данные необходимо ввести правильно. В этом случае для выбора нужного варианта работы процедуры целесообразно использовать значение параметра с именем кнопки Submit – он обычно существует в параметрах запроса только в том случае, когда вызов модуля осуществляется из формы (если не ввести его искусственно).

Обработка данных из форм и внесение их в базу запросами sql является одной из ключевых точек уязвимости приложений (см. sql-injection). Поэтому, в подобных случаях используют:

1. Фильтрацию получаемых данных, избавление от ненужных тегов

2. Передачу данных в SQL-запрос через механизм параметров SQL

3. ORM-обертку и доступ к БД на основе объектов.

С помощью ORM реализуем сохранение данных в таблицу s_emp. Для этого создадим класс Employers и заполним его полями из таблицы s_emp.

Далее, добавим геттеры и сеттеры для полей. Для поля $emp_id будет только геттер, поскольку значение этого поля устанавливается автоматически при вставке новой строки в БД. Конструктор объявим закрытым. Тогда для создания новых пустых экземпляров будет использовать статический метод newEmpltyInstnace:

```
class Employers {
 public static function newEmptyInstance() {
       return new self();
```

```
    }
           public $emp_id;
    private $name;
    private $surname;
    private $dept_id;
    private function __construct() {
    }
    public function getId() {
          return $this->id;
    }
    public function setName($aName) {
          $this->name = $aName;
    }
   }
```

Так, теперь мы можем создавать новый пустой экземпляр объекта и присваивать его полям значения. Нужно добавить метод сохранения значений в БД:

```
    public function save() {
    if (isset($this->id)) {
          $this->_update();
    } else {
          $this->_insert();
    }}
   private function _update() {
    mysql_query("UPDATE `s_emp` SET `name`='{$this-
>name}', "
          . "`surname`='{$this->surname}',
`dept_id`='{$this->dept_id}");}
   private function _insert() {
    mysql_query("INSERT INTO `s_emp` (`name`,
`surname`, `dept_id`)"
          . " VALUES ('{$this->name}', '{$this-
>surname}', '{$this->dept_id}')");
    $new_id = mysql_insert_id();
    $this->id = $new_id;}
```

Теперь у нас есть весь набор средств для манипуляции таблицей s_emp, используя класс Employers. Поля класса объявлены сокрытыми для того, чтобы нельзя было напрямую устанавливать значения, поскольку в сеттерах полей мы

можем проверять устанавливаемые значения в соответствии в типом данных, указанным в БД для исключения ошибок.

5.11. Контрольные задания

5.11.1. Варианты заданий для реализации

1. Система документооборота
2. Приложение стоматологическая клиника
3. Система шиномонтажа
4. Сайт кинотеатра
5. Приложение покупки билетов в кинотеатре
6. Приложение для боулинг-клуба
7. Приложение для гольф-клуба
8. Система управления персоналом
9. Система для управления заводом запчастей
10. Система учета медикаментов

5.11.2. Вопросы для самопроверки

1. Для чего используются HTTP-заголовки?
2. Перечислите основные HTTP-заголовки.
3. Какие переменные окружения используются в заголовоке GET?
4. Из каких этапов состоит создание формы?
5. Этапы проверки корректности данных.
6. Функции для работы с БД в PHP.
7. Какие виды атак существуют?
8. Как избежать взломов и атак?
9. Как осуществляется загрузка файла на сервер?
10. Перечислите функции для работы с изображениями.
11. Как в PHP реализовать отправку e-mail?
12. Что такое JSON и AJAX?
13. Язык XML.

6. МИКРОФРЕЙМВОРКИ

6.1. Общие особенности

Микрофреймворк – это "каркас" приложения, включающий в себя какие-то роутеры и типовые задачи, которые уже решены. Ключевая часть слова здесь – "микро". Очень облегченный, очень маленький, очень быстрый продукт.

Внутри есть, как правило, роутер, который определяет URL, базовая реализация MVC, модели, контроллеры, обработки HTTP-запросов, обработка ошибок. Это самый минимум, который необходим, чтобы быстро запустить приложение.

Ключевая особенность – это именно маленький и быстрый. Нет ничего лишнего. Он очень быстро работает, очень хорошо выглядит для разработчика. Отлично интегрируются сторонние компоненты туда: он набивается по принципу конструктора. Туда набивается все, что необходимо, и быстро запускается.

Не требует длительного изучения и погружения в код, чтобы просто написать приложение.

Как правило, при разработке больших сложных систем очень многие узлы должны быть показаны заказчику перед тем, как начнется их разработка. С помощью микрофреймворков можно очень быстро и просто создать прототип того функционала, который вы будете реализовывать.

В микрофреймворках нет избыточности. Люди, которые их разрабатывают, взяли и положили туда именно то, что необходимо.

Многие микрофреймворки требуют определенной версии PHP. Как правило, это больше 5.3, потому что там активно используется кодогенерация, пространства имен для разделения компонентов.

Главная задача микрофреймворков – это простое расширяемое ядро. В них нет абстрактного уровня базы данных, нет валидации форм или всего того, что уже реализовано в различных сторонних библиотеках, к которым можно обращаться. Микрофреймворк всегда можно расширить самостоятельно и добавить необходимую функциональность. При этом, так как микрофреймворк предлагает только базовый функционал – он очень прост в работе и изучении.

1. Slim – это микрофреймворк позволяющий выполнять все основные задачи. Список возможностей фреймворка:

–Powerful router
–Template rendering with custom views
–Flash messages
–Secure cookies with AES-256 encryption
–HTTP caching
–Logging with custom log writers
–Error handling and debugging
–Middleware and hook architecture
–Simple configuration

Slim — микрофреймворк, идеально подходящий для небольших проектов или приложений, где полноценный фреймворк покажется лишним. Его используют многие PHP-разработчики для создания RESTful API и сервисов. Среди функций Slim — кэширование HTTP на стороне клиента, URL-маршрутизация, шифрование сессий и cookie, а также мгновенные сообщения по HTTP-запросам. Документация полная и сделана качественно.

Вы можете скачать его вручную архивом или воспользоваться Composer.

2. Silex - Для своей работы Silex использует ключевые компоненты Symfony2 в связке с несложной реализацией шаблона проектирования “Внедрение зависимостей” в виде сервис-контейнера Pimple, который позволяет максимально

просто отделить логические части кода друг от друга — сделать их независимыми.

У Silex интуитивно понятный API, что делает процесс разработки достаточно приятным и позволяет практически в несколько шагов добавлять свой функционал в сам фреймворк.

Для работы фреймворка понадобится PHP версии не ниже 5.3 с установленными в php.ini

phar.readonly = Off

phar.require_hash = Off .

Установка фреймворка сводится к загрузке phar-архива и подключении его в bootstrap файле (index.php). Стоит упомянуть, что все запросы нужно перенаправлять именно на этот файл.

Silex позволяет принимать маршруты неограниченной длины.В маршрутах можно использовать переменные.Запросы могут быть POST, PUT, DELETE. К каждому маршруту можно прицепить цепочку валидаторов, А также присвоить название.

Перед и после каждого запроса можно исполнять произвольный код с помощью соответствующих before/after фильтров.

Реализация “Внедрения зависимостей” в виде Pimple является достаточно простой и понятной.

В фреймворке реализована гибкая система расширений с возможностью написания собственных.

3. Fat Free Framework – это микро-фреймворк PHP, выпущенный Bong Cosca в 2009 году. Созданный согласно принципам минимализма, данный фреймворк избегает внесения ненужных кодов и структур и сосредоточен на действительно важных задачах.

Первый шаг – загрузить фреймворк и распаковать файл в root-каталоге нового проекта. FatFree работает только на PHP 5.3 и более новыми версиями.

Создайте файл «index.php», который будет использоваться как загрузочный файл нового проекта. В первую строку внесите Fat Free:

```
// FatFree framework
$f3 = require ("fatfree/lib/base.php");
```

Теперь нужно сказать приложению, в каком режиме (разработки или производства) находится проект, установив эту переменную:

```
// Set to 1 when in development mode,
otherwise set to 0
$f3->set('DEBUG', 1);
```

Конечно же, нужно установить подключение к базе данных. При условии, что используется MySQL:

```
// MySql settings
$f3->set('DB', new DB\SQL(
'mysql:host=localhost;port=3306;dbname=mydata
base',
'dbuser',
'dbpassword'));
```

Чтобы отправить простой запрос, введите:

```
$result = $db->exec('SELECT field FROM table
WHERE id = "1"');
```

При желании можно также использовать встроенный в FatFree ORM.

При помощи функции «DB\SQL\Mapper» можно преобразовать таблицу, которая уже находится в базе данных. Отредактировать таблицу с помощью ORM нельзя.

Поскольку FatFree является микро-фреймворком, в нем нет готовой к использованию структуры проекта, потому ее нужно создать самостоятельно.

Во избежание внесения всех классов в проект FatFree обладает функцией автозагрузки, что позволяет включать классы только тогда, когда они действительно нужны.

Следующий момент, с которым нужно ознакомиться – управление маршрутизацией приложений в FatFree. Маршрутизация к домашней странице определяется так:

```
$f3->route('GET /',
function() {
echo 'ThisismyHomePage!';});
```

Обратите внимание на атрибут *GET*. При необходимости его можно заменить на POST, или даже на GET|POST.

FatFree предоставляет возможность создавать шаблоны и представления.

4**.** Lumen - это микрофреймворк, построенный на основе компонентов Laravel и это официальный микрофреймворк от Laravel. Lumen является одним из самых быстрых микрофреймворков.

Однако, в отличие от многих других микрофреймворков, Lumen позволяет использовать возможности Laravel, такие как роутинг, Внедрение зависимостей, Eloquent ORM, миграции, очереди и даже планировщик команд (scheduledcommands).

Laravel уже быстрый и мощный, но в Lumen убраны многие надстройки, которые предоставляет Laravel.

Lumen включает многие возможности полноценногофреймворкаLaravel:

- Шаблонизатор Blade
- Кэширование
- Планировщик команд (Command Scheduler)
- Контроллеры
- Eloquent ORM
- Управление ошибками
- Абстракция базы данных (Database Abstraction)
- Внедрение зависимостей (Dependency Injection)

- Логгирование
- Очереди

Из-за использования уникального процесса загрузки, Lumen способен обеспечить надежный набор возможностей, и в то же время, предоставить невероятно высокую производительность, что делает его идеальным решением для микро-сервисов PHP.

6.2. Лабораторная работа №6. Разработка приложения MVC

6.2.1 Постановка задачи

Целью лабораторной работы является знакомство со схемой разделения данных приложения MVC. Основное назначение этой концепции состоит в отделении бизнес-логики от её визуализации. За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

Примерный ход выполнения работы.

Устанавливаем фреймворк Silex через утилиту Composer с помощью командной строки:

```
composer create-project fabpot/silex-skeleton lab ~2.0
```

В текущей рабочей директории создается папка lab, куда загрузится демо-приложение.

Для того чтобы перейти в данную папку необходимо ввести команду: `cd lab`.

Далее нам необходимо запустить встроенный PHP-сервер через Composer, предварительно настроив время ожидание процесса на бесконечность:

```
set "COMPOSER_PROCESS_TIMEOUT=0"
composer run
```

Затем проверям результат настроек, запустив в браузере ссылку http://localhost:8888/index_dev.php/

Если ошибок нет, мы увидим страницу-приветствие с текстом "Welcome to your new Silex Application!", а внизу debug-панель (рис. 6. 1):

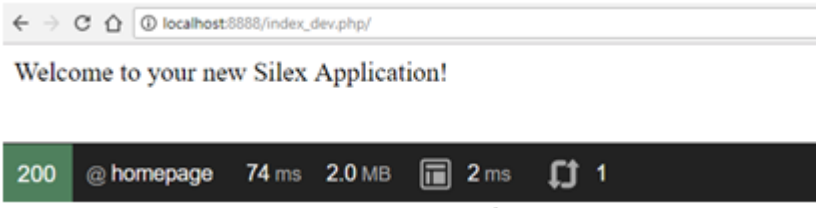

Рисунок 6.1. Страница приветствие

После настройки проекта можно перейти к разработке проекта.

Создаем базу данных - список сотрудников с указанием, работает ли он в данный момент или уволен. Выполним следующий SQL- запрос:

```
CREATE TABLE IF NOT EXISTS `employee` (
`id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `status` enum('active','inactive') NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
INSERT INTO `employee` (`id`, `name`, `status`)
VALUES (1, 'Linus Torvalds', 'active'),
(2, 'Eugene Kaspersky', 'inactive'),
(3, 'Bill Gates', 'active'),
(4, 'Steve Jobs', 'inactive'),
(5, 'Sid Meier', 'active');
ALTER TABLE `employee` ADD PRIMARY KEY (`id`);
ALTER TABLE `employee` MODIFY `id` int(11) NOT
NULL AUTO_INCREMENT;
```

Для того, чтобы подключаться к базе данных, будем использовать библиотеку Doctrine DBAL, устанавливаем её в папке проекта lab с помощью команды:

```
composer require doctrine/dbal:~2.2
```

Фреймворк Silex уже содержит готовый класс для интеграции с данной библиотекой. Также, чтобы работать с базой MySQL, в файле php.ini следует включить данное расширение, раскомментировав строчку:

```
extension=php_pdo_mysql.dll
```

Добавим в файл config/prod.php следующие строки:

```
$app['db.options'] = array_merge([
    'driver'    => 'pdo_mysql',
    'dbname'    => 'БД',
    'host'      => 'localhost',
    'user'      => 'ЛОГИН',
    'password'  => 'ПАРОЛЬ',
    'charset'   => 'utf8'
], isset($app['db.options']) ? $app['db.options']
: []);
$app->register(new
Silex\Provider\DoctrineServiceProvider());
```

Впишите свои параметры бд, логин и пароль. Здесь мы определили параметры для production-окружения и зарегистрировали класс для работы с БД в приложении.

Также мы воспользовались функцией слияния массивов array_merge, это необходимо для тех случаев, когда настройки на dev-окружении отличаются. К примеру, могут быть другие логин и пароль.

Теперь создадим наш обработчик - будем выводить список активных или уволенных сотрудников, в зависимости от передаваемого статуса в GET-параметре. Добавляем в контроллер src/controllers.php следующие строки:

```
    $app->get('/employee/{status}', function
($status) use ($app) {
        $sql = 'SELECT * FROM employee WHERE status =
?';
        $employeeList = $app['db']->fetchAll($sql,
[$status]);
        return $app['twig']-
>render('employees.html.twig', [
            'status'    => $status,
            'employees' => $employeeList,]);})
```

Команда ->assert('status', 'active|inactive') # показывает допустимые значения для статуса. Строчка value('status', 'active') # показывает статус по умолчанию, когда сам GET-параметр не задан.

Сначала мы формируем SQL-запрос с подстановкой параметра (статус), далее обращаемся к базе - производим выборку всех записей в ассоциативный массив.

После этого через шаблонизатор Twig получаем HTML-представление. Для этого создаем соответствующий файл templates/employees.html.twig и заполняем его:

```
    {% extends "layout.html.twig" %}
    {% block content %}
        <h1>{{ status == 'active' ? 'Работают' :
'Уволенные' }}</h1>
        <table>
            <thead>
            <tr>
                <th>№</th>
                <th>Имя</th>
            </tr>
            </thead>
            <tbody>
            {# выводим весь список сотрудников #}
            {% for employee in employees %}
                <tr>
                    <td>{{ employee.id }}</td>
                    <td>{{ employee.name }}</td>
```

```
            </tr>
        {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

Первым параметром мы передаем адрес шаблона, а вторым - массив параметров, которые будут доступны в данном шаблоне по имени - ключам массива. Через конструкцию extends шаблон рендерит сначала более общий шаблон layout.html.twig, в котором заменяет блок content через конструкцию block, а далее - содержимое этого блока.

В примере использованы тернарный оператор (?:), цикл ({% for ... in ... %}), вывод ({{ }}) и комментарий ({# #}).

Улучшим визуальное отображение шаблона. Для этого, в общем шаблоне templates/layout.html.twig можно посмотреть путь по которому подключен CSS-файл, который находится по адресу web/css/main.css. Добавим стилей в css-файл.

```
table { width: 100%; text-align: left;
    border-spacing: 0;}
table td {
    border-top: 1px solid black;}
```

Чтобы проверить результат необходимо пройти по ссылке http://localhost:8888/index.php/employee/inactive. В браузере мы увидем 2-х уволенных сотрудников (рис 6.2):

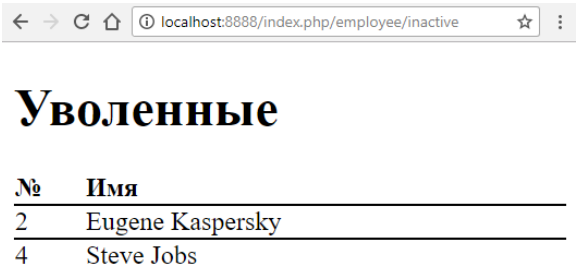

Рисунок 6.2. Уволенные сотрудники

Чтобы просмотреть активных сотрудников необходимо перейти по ссылке:

http://localhost:8888/index.php/employee/active

http://localhost:8888/index.php/employee

В ходе данной лабораторной работы на практике создали полноценное MVC-приложение, работающего с базой данных.

6.3. Контрольные задания

6.3.1. Варианты заданий для реализации

1. Сайт-визитка
2. Сайт для работы с картой города
3. Сайт кинотеатра
4. Сайт поликлиники
5. Сайт полиции
6 Система учета складских помещений.

6.3.2. Вопросы для самопроверки

1. Что такое микрофреймворк?
2. Перечислите ключевые особенности микрофреймворков.
3. Микрофреймворк Slim.
4. Микрофреймворк Silex.
5. Микрофреймворк Fat-free.
6. Микрофреймворк Lumen.

7. СИСТЕМЫ УПРАВЛЕНИЯ КОНТЕНТОМ (CMS)

7.1. Общие особенности

Система управления содержимым — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления содержимым.

Основные функции CMS:

- Предоставление инструментов для создания содержимого, организация совместной работы над содержимым;
- Управление содержимым: хранение, контроль версий, соблюдение режима доступа, управление потоком документов и т. п.;
- Публикация содержимого,
- Представление информации в виде, удобном для навигации, поиска.

В системе управления содержимым могут находиться самые различные данные: документы, фильмы, фотографии, номера телефонов, научные данные и так далее. Такая система часто используется для хранения, управления, пересмотра и публикации документации. Контроль версий является одним из основных её преимуществ, когда содержимое изменяется группой лиц.

Система управления — программа, предоставляющая инструменты для добавления, редактирования, удаления информации на сайте.

Большинство современных CMS имеют модульную архитектуру, что позволяет администратору самому выбирать и настраивать те компоненты, которые ему необходимы. Типичные модули:

- динамическое меню;
- блог;
- новости;

- опросы;
- поиск по сайту;
- статистика посещений;
- гостевая книга и т. д.

Сайты, организованные посредством системы управления контентом, основаны на следующих технологиях: веб-сервер, хранилище данных (зачастую СУБД, например такие как MySQL или PostgreSQL, однако существуют и noSQL CMS), веб-приложение для обеспечения работы самой системы, визуальный редактор страниц, файловый менеджер с веб-интерфейсом для управления файлами сайта, система управления правами пользователей и редакторов сайта.

Существуют разнообразные системы управления сайтом, среди которых встречаются платные и бесплатные, построенные по разным технологиям. Каждый сайт имеет панель управления, которая является только частью всей программы, достаточной для управления сайтом.

Наиболее распространены следующие технологические платформы, используемые в качестве основы веб-приложения, реализующего работу CMS: PHP, Perl, .NET.

Существует термин контент-менеджер, обозначающий род профессиональной деятельности — редактор сайта или сотрудника, работающего с CMS.

Большая часть современных систем управления содержимым реализуется в виде визуального редактора — программы, которая создаёт HTML-код из специальной упрощённой разметки, позволяющей пользователю проще форматировать текст.

1. WordPress — система управления содержимым сайта с открытым исходным кодом, написанная на PHP, сервер базы данных — MySQL. Сфера применения — от блогов до достаточно сложных новостных ресурсов и интернет-магазинов. Встроенная система «тем» и «плагинов» вместе с

удачной архитектурой позволяет конструировать проекты широкой функциональной сложности.

Преимущества:

- Бесплатность. WordPress — это бесплатная система. Для новичка, который хочет создать свой блог или небольшой проект, это немаловажный момент и огромное преимущество;

- Простота установки и использования. Весь процесс установки занимает не более 5-ти минут, и для этого не нужно быть программистом, разбираться в коде и технических нюансах;

- Кроссплатформенность. WordPress устанавливается и используется непосредственно на вашем сайте (сервере). На компьютер не нужно ничего дополнительно устанавливать. Это значит, что вы можете управлять своим сайтом с любого компьютера из под любой операционной системы. Единственное необходимое условие — это подключение к Интернету;

- Встроенный редактор. Пользоваться WordPress-ом очень просто и легко в основном благодаря интуитивно понятному встроенному редактору;

- Популярность. WordPress — это самая популярная в мире система управления содержимым сайта;

- огромная библиотека качественных платных тем, которые обладают более продвинутым функционалом и гибкостью настройки;

- поддержка виджетов и социальных плагинов для улучшения читаемости и посещаемости вашего ресурса;

- надежность и безопасность системы от взлома;

- встроенная поддержка системы поисковой оптимизации (SEO) — незаменимой составляющей для повышения рейтинга сайта в поисковых системах;

- реализованная поддержка взаимодействия с социальными сетями и RSS лентой новостей. Удобная система комментирования поможет получить адекватную и быструю реакцию на публикуемый материал. По желанию можно

подключить стороннюю систему комментирования, например, от Facebook или Disqus, которые обладают своими преимуществами.

2. Joomla — система управления содержимым (CMS), написанная на языках PHP и JavaScript, использующая в качестве хранилища базы данных СУБД MySQL или другие стандартные промышленные реляционные СУБД. Является свободным программным обеспечением, распространяемым под лицензией GNU GPL.

CMS Joomla включает в себя минимальный набор инструментов при начальной установке, который дополняется по мере необходимости. Это снижает загромождение административной панели ненужными элементами, а также снижает нагрузку на сервер и экономит место на хостинге.

Joomla позволяет отображать интерфейс фронтальной и административной части на любом языке. Каталог расширений содержит множество языковых пакетов, которые устанавливаются штатными средствами администрирования. Доступны пакеты русского, украинского, белорусского и ещё некоторых языков пост-советского пространства.

Возможности:

- Функциональность можно увеличивать с помощью дополнительных расширений (компонентов, модулей и плагинов);
- Имеется модуль безопасности для многоуровневой аутентификации пользователей и администраторов (используется собственный алгоритм аутентификации и «ведения» сессий);
- Система шаблонов позволяет легко изменять внешний вид сайта: расположение модулей, шрифты и другое. Можно предоставить пользователям выбирать одно из нескольких отображений. В сети существует огромный выбор готовых шаблонов, как платных, так и бесплатных. Также существует программное обеспечение для самостоятельного создания оригинальных шаблонов;

- Предусмотрены настраиваемые схемы расположения модулей, включая левый, правый, центральный и любое другое произвольное положения блока. При желании содержимое модуля можно включить в содержимое материала;

- К преимуществам системы можно отнести то, что все компоненты, модули, плагины и шаблоны можно написать самому, разместить их в структурированном каталоге расширений или отредактировать существующее расширение по своему усмотрению;

Происходит регулярный выход обновлений. Существует публичный «баг-трекер» (система отслеживания ошибок).

3. Drupal — система управления содержимым (CMS), используемая также как каркас для веб-приложений (CMF), написанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных (поддерживаются MySQL, PostgreSQL и другие). Drupal является свободным программным обеспечением, защищённым лицензией GPL, и развивается усилиями энтузиастов со всего мира.

Архитектура Drupal позволяет применять его для построения различных типов сайтов — от блогов и новостных сайтов до информационных архивов или социальных сетей. Имеющуюся по умолчанию функциональность можно увеличивать подключением дополнительных расширений — «модулей» в терминологии Drupal. Наиболее важные функции, предоставляемые Drupal:

- единая категоризация всех видов содержимого (таксономия) — от форумных сообщений до блогов и новостных статей;

- широкий набор свойств при построении рубрикаторов: плоские списки, иерархии, иерархии с общими предками, синонимы, родственные категории;

- вложенность категорий любой глубины;

- поиск по содержимому сайта, в том числе поиск по таксономии и пользователям;

- разграничение доступа пользователей к материалам (ролевая модель);
- динамическое построение меню;
- поддержка XML-форматов;
- вывод документов в RDF/RSS;
- агрегация материалов с других сайтов;
- BlogAPI для публикации материалов с помощью внешних приложений;
- переводы интерфейса сайта на разные языки, а также поддержка ведения разноязычного контента;
- возможность создания сайтов с пересекающимся содержимым;
- раздельные конфигурации сайта для различных виртуальных хостов (мультисайтинг), в том числе собственные наборы модулей и тем оформления для каждого подсайта;
- уведомления о выходящих обновлениях модулей;

В Drupal предлагается гибкая схема организации структуры сайта на основе таксономии. Таксономия — механизм, позволяющий создавать произвольное количество тематических категорий для содержимого сайта и ассоциировать их с модулями, обеспечивающими ввод и вывод информации. Категории могут представлять плоские или иерархические списки, либо сложные структуры, где элемент может иметь несколько «родителей» и несколько дочерних элементов. С помощью подобной схемы одними и теми же модулями возможна организация различных вариантов структуризации содержимого. Например, легко создаётся сквозной список «ключевых слов» для всех документов сайта и т. п.

Другая парадигма появилась с созданием в Drupal расширения Content Construction Kit (CCK). CCK позволяет дополнять документы новыми полями различных типов — от полей ввода URL и email, до полей хранения и отображения мультимедийных файлов. Также посредством дополнительных

модулей к CCK можно организовать связи между документами, не используя механизм таксономии.

Drupal имеет модульную архитектуру с компактным ядром, предоставляющим API, к которому могут обращаться модули. Стандартный набор модулей включает такие функции, как новостная лента, блог, форум, загрузка файлов, сборщик новостей, голосования, поиск и другие. Дизайн сайта меняется также посредством специальных модулей — «тем оформления».

4. TYPO3 — система управления сайтами (CMS/CMF) с открытым исходным кодом и свободной лицензией. Является гибкой расширяемой системой с большим количеством модулей и функций. Написана на PHP, для хранения данных использует любую реляционную базу данных, поддерживаемую TYPO3 DBAL, включая MySQL, Oracle Database, PostgreSQL и другие. Работает на таких серверах, как Apache или IIS, и на большинстве операционных систем. Система создана Каспером Скорхёем и распространяется бесплатно под лицензией GNU GPL.

Структура сайта в TYPO3 представлена деревом страниц. На каждой странице могут быть размещены элементы содержимого — небольшие блоки информации: текст, текст + изображение, изображение, таблица, чистый html, плагин и др. TYPO3 основана на шаблонах. Существуют готовые шаблоны для TYPO3, однако в основном шаблоны строятся заново на специальном конфигурационном языке TypoScript. Несмотря на использование script в названии, TypoScript не является процедурным языком. Он используется для конфигурирования и отображения сайта. TypoScript является альтернативой другим конфигурационным языкам, таким как ini- или conf-файлы, XML или JSON.

Основные возможности:

- Редактор текста с форматированием (RTE) и проверка орфографии;

- Редактор текста с форматированием (FCKeditor) и модификация от сторонних разработчиков;
- Отмена изменений и история изменений;
- Предпросмотр перед публикацией;
- Буфер обмена;
- Редактирование содержимого из frontend и backend;
- Внутренние ссылки (возможны ссылки на уровне элементов содержимого);
- Внутренний поисковый механизм;
- Одна установка системы для многих сайтов (с единым деревом страниц);
- Одна установка системы для многих доменов;
- Гибкая система разграничения прав для редакторов сайта;
- Динамическое построение текстовых и графических меню;
- Возможность авторизации пользователей через LDAP.

Templavoila — альтернативный шаблонизатор для TYPO3. С помощью Templavoila из HTML-шаблона генерируется TYPO3-шаблон без изменений в структуре HTML. Процесс генерации Templavoila-шаблона (мэппинг) представляет собой сопоставление областей HTML-кода и той функциональности, которая будет связана с этими областями. Например, один из тегов будет заменяться на меню, вместо другого тега будет подставлен элемент контента. Мэппинг выполняется с помощью мыши и не требует даже знания HTML.

Важной особенностью Templavoila является возможность создания Flexible Content Elements (FCE) — динамических элементов контента. Это особые элементы контента, которые могут включать в себя более мелкие элементы контента и располагать их в соответствии со структурой HTML-шаблона. FCE строятся по тем же принципам, что и Templavoila-шаблоны страниц, создавая эффект «шаблона в шаблоне».

Использование FCE позволяет чрезвычайно гибко управлять содержимым страницы.

7.2. Лабораторная работа №7. Разработка Single Page Application

7.2.1. Постановка задачи

Цель работы является знакомство с модульными одностраничными приложениями с подгрузкой информации через Ajax. Самая заметная разница между обычным сайтом и SPA — это уменьшение количества обновлений страницы. У SPA более тяжелое использование Ajax — способ общения с серверными серверами без полного обновления страницы — для загрузки данных в наше приложение. В результате процесс построения страниц происходит в основном на стороне клиента с помощью JavaScript.

Примерный ход выполнения работы

Для того, чтобы использовать сторонние JS и CSS библиотеки, воспользуемся менеджером пакетов Bower.

Для начала, находясь в папке проекта lab, создадим bower.json с помощью команды:

bower init

На появляющихся предложениях ввода - просто прощелкиваем Enter, оставляя значения по умолчанию.

Теперь установим фреймворк Bootstrap:

bower install bootstrap

Он включает в себя и готовые файлы-стили, и свежую версию библиотеки jQuery.

Обратите внимание, скаченные с git-репозитория файлы библиотеки расположились в папке bower_components.

Теперь нужные нам файлы необходимо перенести в нашу рабочую папку web.

Для начала создаем папки css и js в папке web, если их нет. А далее копируем в них файлы:

```
    copy bower_components\jquery\dist\jquery.min.js
web\js\jquery.js
    copy
bower_components\bootstrap\dist\css\bootstrap.min.css
web\css\bootstrap.css
```

Теперь данные файлы необходимо включить на страницу. Добавляем в файл templates/layout.html.twig в тег

head - `<link href="{{ asset('css/bootstrap.css') }}" rel="stylesheet" type="text/css" />`

в конец body - `<script type="text/javascript" src="{{ asset('js/jquery.js') }}"></script>`

Создадим таблицу БД с данными, используя следующий скрипт:

```
    CREATE TABLE IF NOT EXISTS `task` (
    `id` int(11) NOT NULL,
      `text` varchar(256) NOT NULL,
      `status` enum('new','completed') NOT NULL
DEFAULT 'new'
    ) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
    INSERT INTO `task` (`id`, `text`, `status`)
VALUES
    (1, 'Прочитать методические указания',
'completed'),
    (2, 'Получить задание у преподавателя', 'new'),
    (3, 'Ознакомиться с текстом лабораторной работы',
'new'),
    (4, 'Выполнить лабораторную работу', 'new');
    ALTER TABLE `task` ADD PRIMARY KEY (`id`);
    ALTER TABLE `task` MODIFY `id` int(11) NOT NULL
AUTO_INCREMENT;
```

На нашей странице будет 2 основных блока:

- количество выполненных заданий из общего числа, который будет постоянно обновляться

- поручением с 2-мя кнопками: исполнения и перехода к следующему

Все запросы будут осуществляться через AJAX с использованием jQuery.

Начинаем с определения контроллера (src/controllers.php):

```
    $app->get('stat', function () use ($app) {
        $sql = 'SELECT COUNT(*) AS count FROM task
UNION SELECT COUNT(*) FROM task WHERE status =
"completed"';
        $result = $app['db']->fetchAll($sql);
        return $app->json([
            'total'     => $result[0]['count'],
            'completed'    =>    isset($result[1])    ?
$result[1]['count']   :   $result[0]['count'],   #   с
обработкой на случай всех выполненных поручений
        ]);});
    $app->get('task/next',      function      (Request
$request) use ($app) {
        $getParams    =    $request->query->all();    #
получаем GET-параметры
        $after  =  isset($getParams['after'])  ?  (int)
$getParams['after'] : 0;
        $sql = 'SELECT * FROM task WHERE status =
"new" AND id > ? LIMIT 1';
    $task = $app['db']->fetchAssoc($sql, [$after]);
        return $app['twig']-
>render('task/item.html.twig', [
            'id'    => $task ? $task['id'] : 0,
            'text'  => $task ? $task['text'] : '',
        ]);
    });

    $app->post('task/{id}/complete', function ($id)
use ($app) {
        $sql = 'UPDATE task SET status = "completed"
WHERE id = ?';
        $id = $app['db']->executeUpdate($sql, [(int)
$id]);
        if (!$id) {
```

```
            $app->error(function () { return new
Response('Ошибка обновления БД'); });
        }
        return $app->json(['id' => $id]);
    })
    ->assert('id', '\d+') # принимаем только цифры
```

В первом у нас возвращается JSON-ответ с цифрами, второй рендерит готовый html-кусок, третий через POST-запрос проставляет поручение выполненным.

Определяем шаблон для поручения (templates/task/item.html.twig):

```
    <div class="jumbotron">
        <h2>Поручение</h2>
        {% if id|default(0) > 0 %}
        <p>{{ text }}</p>
        <p>
            <a id="complete" class="btn btn-success
btn-lg" data-id="{{ id }}">Выполнить</a>
            <a id="next" class="btn btn-primary btn-
lg" data-id="{{ id }}">Следующее</a>
        </p>
        {% else %}
            <p>{{ text|default('Поручений больше
нет!') }}</p>
        {% endif %}
    </div>
```

Через фильтр default можно определять значения, если они не заданы для шаблона.

Далее правим наш общий шаблон (templates/layout.html.twig) в соответствии с кодом:

```
    <!DOCTYPE html>
    <html>
        <head>
            <title>Lab #7</title>
```

```
            <link href="{{
asset('css/bootstrap.css') }}" rel="stylesheet"
type="text/css" />
        </head>
        <body style="margin-top: 20px">
            <div class="container">
                {% block content %}{% endblock %}
            </div>
            <script type="text/javascript"
src="{{ asset('js/jquery.js') }}"></script>
            {% block js %}{% endblock %}
        </body>
    </html>
```

Обратите внимание на блок js - здесь мы определим свои скрипты для AJAX-запросов.

И, наконец, формируем шаблон главной страницы (templates/index.html.twig):

```
    {% extends "layout.html.twig" %}
    {% block content %}
        <div class="well">Выполнено <span
id="completed_task_count">?</span> из <span
id="total_task_count">?</span> поручений</div>
        <div id="task">
            {% include 'task/item.html.twig' with
{text: '...'} %}
        </div>
    {% endblock %}
    {% block js %}
        <script type="text/javascript" src="{{
asset('js/main.js') }}"></script>
    {% endblock %}
```

Здесь использована директива include, позволяющая подключать из шаблона другой шаблон с передачей параметров в JSON-формате.

Также мы подключили свой файл скрипта (web/js/main.js), заполним и его:

```
    setInterval(() => {
        $.get('stat', (data) => {

$('#completed_task_count').html(data.completed)

$('#total_task_count').html(data.total)
        })
    }, 1000)
    let getNextTask = (afterId) => {
        $.get('task/next', {after: afterId},
(data) => {
            $('#task').html(data)
        })}
    $('#task').on('click', '#next', function () {
        const id = $(this).data('id') // выбирем
номер текущего поручения из data-атрибута
        getNextTask(id) // делаем запрос на
следующее
    })

    $('#task').on('click', '#complete', function
() {
        const id = $(this).data('id')
        $.post('task/' + id + '/complete', (data)
=> {
            getNextTask(data.id)
        })
    })
getNextTask(0)
```

Здесь мы определяем действие по таймеру - ежесекундное обновление статистики. Далее функцию, которая AJAX-запросом получает следующее поручение (принимает параметр - номер предыдущего поручения). Далее мы создаем обработчики кликов на кнопки со своими

действиями. И в конце вызываем функцию для получения первого поручения.

Теперь посмотрим, что у нас должно получиться. Заходим на страницу http://localhost:8888/index_dev.php/ и увидим следующее:

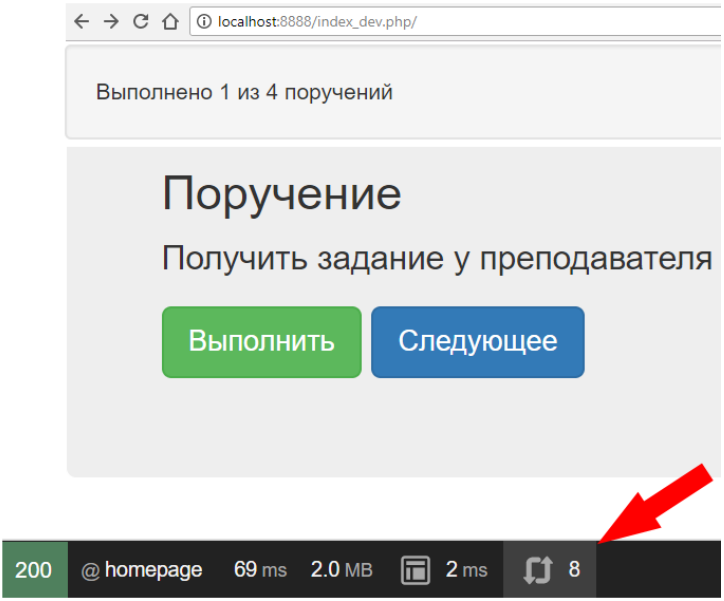

Рис. 6.1. Выполненные поручения

Стрелочка указывает на debug-панель, в которой происходит считывание всех запросов со страницы. AJAX-запросы там также регистрируются. Поэтому если ничего не нажимать, то ежесекундно число будет увеличиваться.

Убеждаемся, что все кнопки рабочие. Для того, чтобы заново вернуться к невыполненным заданиям - необходимо перезагрузить страницу (Ctrl + F5).

Так как обновление статистической инфы происходит раз в секунду, то изначально вместо чисел пользователь увидит

знаки вопроса, которые потом заменяться на данные из AJAX-запроса. И при выполнении всех поручений до первой секунды главная страница будет выглядеть так (рис. 6.2):

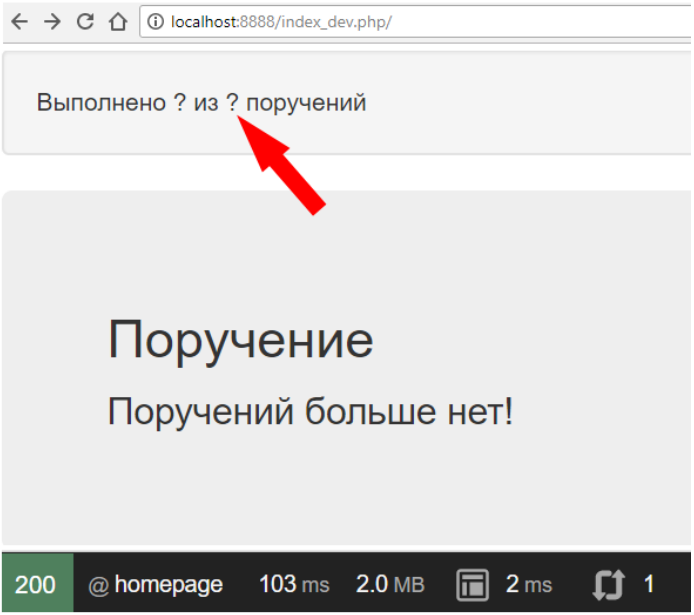

Рис.6.2. Выполненные поручения

В ходе данной лабораторной работы создали полноценное одностраничное приложение, построенное на MVC-архитектуре по модульному принципу. Информация для каждого модуля подгружается через AJAX-запросы с использованием библиотеки jQuery.

7.3. Контрольные задания

7.3.1. Варианты заданий для реализации

1. Интернет-издание

2. Почтовая служба
3. Веб-сервис для поиска иконок
4. Одностраничный интернет-магазин
5. Сервис электронных платежей

7.3.2. Вопросы для самопроверки

1. Системы управления контентом (CMS).
2. Система управления WordPress.
3. Преимущества WordPress.
4. Возможности системы Joomla.
5. Назовите самые важные функции Drupal.
6. Предназначение расширения Content Construction Kit.
7. Возможности TYPO3.
8. Что такое Templavoila?
9. Особенности Templavoila.

8. АСИНХРОННЫЕ ПРИЛОЖЕНИЯ В PHP

8.1. SOA

Сервис-ориентированная архитектура (SOA) — модульный подход к разработке программного обеспечения, основанный на использовании распределённых, слабосвязанных заменяемых компонентов, оснащённых стандартизированными интерфейсами для взаимодействия по стандартизированным протоколам.

Программные комплексы, разработанные в соответствии с сервис-ориентированной архитектурой, обычно реализуются как набор веб-служб, взаимодействующих по протоколу SOAP, но существуют и другие реализации.

Интерфейсы компонентов в сервис-ориентированной архитектуре инкапсулируют детали реализации (операционную систему, платформу, язык программирования) от остальных компонентов, таким образом обеспечивая комбинирование и многократное использование компонентов для построения сложных распределённых программных комплексов, обеспечивая независимость от используемых платформ и инструментов разработки, способствуя масштабируемости и управляемости создаваемых систем.

Архитектура не привязана к какой-то определённой технологии. Она может быть реализована с использованием широкого спектра технологий, включая такие технологии как REST, RPC, DCOM, CORBA или веб-сервисы. SOA может быть реализована, используя один из этих протоколов и, например, может использовать дополнительно механизм файловой системы для обмена данными.

Главное, что отличает SOA — это использование независимых сервисов с чётко определёнными интерфейсами, которые для выполнения своих задач могут быть вызваны

неким стандартным способом, при условии, что сервисы заранее ничего не знают о приложении, которое их вызовет, а приложение не знает, каким образом сервисы выполняют свою задачу.

SOA также может рассматриваться как стиль архитектуры информационных систем, который позволяет создавать приложения, построенные путём комбинации слабосвязанных и взаимодействующих сервисов. Эти сервисы взаимодействуют на основе какого-либо строго определённого платформенно-независимого и языково-независимого интерфейса (например, WSDL). Определение интерфейса скрывает языково-зависимую реализацию сервиса.

Таким образом, системы, основанные на SOA, могут быть независимы от технологий разработки и платформ.ьК примеру, сервисы, написанные на C#, работающие на платформах .Net и сервисы на Java, работающие на платформах Java EE, могут быть с одинаковым успехом вызваны общим составным приложением. Приложения, работающие на одних платформах, могут вызывать сервисы, работающие на других платформах, что облегчает повторное использование компонентов.

SOA может поддерживать интеграцию и консолидацию операций в составе сложных систем, однако SOA не определяет и не предоставляет методологий или фреймворков для документирования сервисов.

Языки высокого уровня, такие как BPEL, или спецификации, такие как WS-CDL и WS-Coordination, расширяют концепцию сервиса, предоставляя метод оркестрации, для объединения мелких сервисов в более обширные бизнес-сервисы, которые, в свою очередь, могут быть включены в состав технологических процессов и бизнес-процессов, реализованных в виде составных приложений или порталов.

Использование компонентной архитектуры (SCA) для реализации SOA — это область текущих исследований.

8.2. Messaging (система мгновенного обмена сообщениями)

Messaging это службы мгновенных сообщений программы онлайн-консультанты и программы-клиенты для обмена сообщениями в реальном времени через Интернет. Могут передаваться текстовые сообщения, звуковые сигналы, изображения, видео, а также производиться такие действия, как совместное рисование или игры. Многие из таких программ-клиентов могут применяться для организации групповых текстовых чатов или видеоконференций. Для подобного рода коммуникации необходима клиентская программа, так называемый мессенджер. Отличие от электронной почты здесь в том, что обмен сообщениями идёт в реальном времени (англ. *instant* — мгновенно). Большинство IM-клиентов позволяет видеть, подключены ли в данный момент абоненты, занесённые в список контактов. В ранних версиях программ всё, что печатал пользователь, тут же передавалось. Если он делал ошибку и исправлял её, это тоже было видно. В таком режиме общение напоминало телефонный разговор. В современных программах сообщения появляются на мониторе собеседника уже после окончания редактирования и отправки сообщения.

Как правило, мессенджеры не работают самостоятельно, а подключаются к центральному компьютеру сети обмена сообщениями, называемому сервером. Поэтому мессенджеры и называют клиентами (клиентскими программами). Термин является понятием из клиент-серверных технологий.

Широкому кругу пользователей известно некоторое количество популярных сетей (и клиентов) обмена сообщениями, таких как IRC, Skype, ooVoo, AIM, ICQ, MSN, Yahoo!, Jitsi, XMPP. Каждая из этих сетей разработана отдельной группой разработчиков, имеет отдельные серверы и протоколы, отличается своими правилами и особенностями. Между

различными сетями обычно нет прямой связи (только в XMPP существует понятие межсетевого транспорта), таким образом, пользователь сети Skype не может связаться с пользователем сети ICQ, однако ничто не мешает быть одновременно пользователем нескольких сетей.

Почти для каждой из сетей есть свой мессенджер, разработанный той же командой разработчиков. Так, для пользования тремя последними из вышеуказанных сетей разработчиками предлагаются программы с одноимёнными названиями: ICQ, Windows Live Messenger, Yahoo! Messenger, а также Skype. Таким образом, если один из адресатов пользуется только сетью ICQ, а другой — только сетью MSN, то можно общаться с ними одновременно, установив на своем компьютере и ICQ, и MSN Messenger и зарегистрировавшись в обеих сетях (либо через соответствующие транспорты в XMPP).

В качестве альтернативного мессенджера можно выбрать программу стороннего производителя, как коммерческую, так и бесплатную. Популярными альтернативными программами для общения в сети ICQ являются QIP 2005/QIP Infium, Psi/Psi+ (через XMPP-транспорт), Trillian, Miranda IM, Pidgin, MyChat. Также некоторые из них позволяют подключаться одновременно к нескольким сетям, то есть являются мультипротокольными, что избавляет от необходимости устанавливать отдельный мессенджер для каждой сети и позволяет общаться со всеми адресатами единым образом независимо от сети; все перечисленные в предыдущем предложении клиенты ICQ, за исключением версии QIP 2005, поддерживают и протокол XMPP.

Большинство IM-сетей используют закрытые протоколы, поэтому альтернативные клиенты теоретически могут обладать меньшим количеством базовых функций, чем официальные, хотя на практике чаще бывает наоборот. Однако при изменениях протокола на стороне сервера сети

альтернативные клиенты могут внезапно перестать работать (например, подобное явление наблюдалось для «нефирменных» клиентов сервиса ICQ в России).

В качестве альтернативы проприетарным протоколам для IM был разработан открытый и хорошо расширяемый протокол XMPP (также известный как Jabber), используемый в таких сервисах, как Google Talk, Я.Онлайн и др. Этот протокол часто используется для организации общения в корпоративных и других локальных сетях и имеет ряд существенных преимуществ, как, например, шифрование сообщений и стабильность на неустойчивых каналах связи. Протокол децентрализованный, его архитектура напоминает электронную почту, где возможно общение между пользователями, имеющими аккаунты на разных серверах. Если нарушится работа одного сервера, то это не повлияет на работу всей сети.

8.3. Системы очередей

Очередь сообщений позволяет обеспечить асинхронное выполнение участков программы. Это позволяет:

- увеличить скорость работы приложения;
- обслуживать большее количество посетителей;
- использовать разные языки разработки в одном приложении.

Система очередей — это принцип, а не конкретная технология. Для реализации системы очередей не обязательно использовать внешнее решение. Вы можете реализовать очередь, скажем, на MySQL и PHP. Однако простота и наличие готовых решений позволят сделать это быстрее.

Использование cron скриптов (например, для преобразования видео файлов) — это самый примитивный метод реализации очередей. Современные системы позволяют сделать все намного проще и удобнее.

Система очередей состоит из двух основных компонент:

- сервер очереди хранит список сообщений , которые отправляет ему основное приложение. Задача — это просто информация о том, что и как нужно выполнить.
- обработчик — это часть основной программы, которая работает с очередью в обратном направлении. Он получает новые сообщения из очереди и выполняет соответствующие действия.

1. RabbitMQ — платформа, реализующая систему обмена сообщениями между компонентами программной системы на основе стандарта AMQP (Advanced Message Queuing Protocol).

RabbitMQ состоит из:

- Сервера RabbitMQ,
- Поддержки протоколов HTTP, XMPP и STOMP,
- Клиентских библиотек AMQP для Java и .NET Framework (поддержка других языков программирования реализована в ПО других производителей)
- Различных плагинов (таких как плагины для мониторинга и управления через HTTP или веб-интерфейс или плагин «Shovel» для передачи сообщений между брокерами)

Поддерживается горизонтальное масштабирование для построения кластерной архитектуры.

Имеется реализация клиентов для доступа к RabbitMQ для целого ряда языков программирования и платформ, широко используемых для веб-разработки: Java, .NET, Perl, Python, Ruby, PHP и других.

В качестве движка базы данных для хранения сообщений используется Mnesia.

2. AMQP (Advanced Message Queuing Protocol) — открытый протокол для передачи сообщений между компонентами системы. Основная идея состоит в том, что отдельные подсистемы (или независимые приложения) могут обмениваться произвольным образом сообщениями через AMQP-брокер, который осуществляет маршрутизацию, возможно гарантирует доставку, распределение потоков данных, подписку на нужные типы сообщений.

AMQP основан на трёх понятиях:

Сообщение (message) — единица передаваемых данных, основная его часть (содержание) никак не интерпретируется сервером, к сообщению могут быть присоединены структурированные заголовки.

Точка обмена (exchange) — в неё отправляются сообщения. Точка обмена распределяет сообщения в одну или несколько очередей. При этом в точке обмена сообщения не хранятся. Точки обмена бывают трёх типов:

fanout — сообщение передаётся во все прицепленные к ней очереди;

direct — сообщение передаётся в очередь с именем, совпадающим с ключом маршрутизации (routing key) (ключ маршрутизации указывается при отправке сообщения);

topic — нечто среднее между fanout и direct, сообщение передаётся в очереди, для которых совпадает маска на ключ маршрутизации, например, app.notification.sms.# — в очередь будут доставлены все сообщения, отправленные с ключами, начинающимися на app.notification.sms.

Очередь — здесь хранятся сообщения до тех пор, пока не будут забраны клиентом. Клиент всегда забирает сообщения из одной или нескольких очередей.

3. Websphere - это платформа интеграционного программного обеспечения (ПО) компании IBM. Она включает в себя полную инфраструктуру промежуточного ПО (такого как серверы, сервисы и инструменты), необходимого для

написания, запуска и мониторинга в режиме 24x7 web-приложений по требованию, а также кроссплатформенных решений и решений, в которые вовлечены различные продукты. WebSphere предоставляет надежное, гибкое и устойчивое интеграционное ПО.

WebSphere Application Server - это основа инфраструктуры; на нем запускается все остальное.

ПО CICS Transaction Server является основой большинства современных приложений на мэйнфреймах. Оно поддерживает разработку программного обеспечения на таких популярных языках, как COBOL, PL/I, C/C++ и Java.

WebSphere представляет собой модульную платформу, построенную на базе открытых стандартов, широко поддерживаемых в отрасли. С помощью надёжных и проверенных интерфейсов вы можете подключить к WebSphere существующие активы, а при необходимости роста вы сможете продолжить развивать вашу среду. WebSphere работает на множестве платформ, в том числе AIX, HP Unix, i5/OS, Linux, Sun Solaris, Windows и z/OS.

Возможности:

1. Бизнес по требованию - это такой бизнес, чьи бизнес-процессы (интегрированные непрерывно внутри компании, а также с ключевыми партнерами, поставщиками и заказчиками) способны быстро реагировать на любые требования пользователя, возможности расширения рынка или внешние угрозы. Вы можете использовать WebSphere для построения и мониторинга инфраструктуры с целью поддержки вашего бизнеса по требованию, а также для создания и расширения приложений, которые запускаются в этой инфраструктуре.

2. Интеграция процессов - возможности интеграции процессов могут использоваться в бизнесе для моделирования, постановки, мониторинга и оптимизации бизнес-процессов в соответствии со стратегическими задачами. К примеру, вы можете смоделировать ключевой бизнес-процесс, затем симулировать его, усовершенствовать, позволить людям

взаимодействовать с ним удобными способами, ввести его в производство, контролировать его работу, точно отрегулировать, а затем, при необходимости, быстро и разумно адаптировать к изменениям бизнеса.

3. Интеграция информации - возможности информационной интеграции позволяют вам создавать непротиворечивый единообразный вид структурированной и неструктурированной информации из несовместимых источников, а также управлять и синхронизировать справочную информацию о продуктах. Например, вы можете создать поиск (в свободном формате) по всем вашим информационным ресурсам, включая web-сайты, реляционные базы данных, файловые системы, конференции, порталы, системы совместной деятельности и системы управления содержимым.

4. Интеграция приложений - возможности интеграции приложений предоставляют широкое многообразие сервисов для поддержки надежных и гибких информационных потоков внутри приложений, которые могут запускаться на различных предприятиях. К примеру, вы можете просто обмениваться сообщениями между двумя приложениями, либо, если требования вашего бизнеса более сложные, создать гибкую сервис-ориентированную архитектуру для поддержки обмена систематической информацией среди широкого многообразия приложений, запускаемых на серверах различных компаний, различных платформах и поддерживающих различные языки.

5. Инфраструктура приложений - инфраструктура приложений WebSphere позволяет вам создавать, развертывать, интегрировать и улучшать новые и существующие приложения. К примеру, вы можете расширить унаследованные приложения и сделать их доступными для Web или среды Java.

8.4. Лабораторная работа №8. Разработка REST-приложения

8.4.1. Постановка задачи

Целью лабораторной работы является знакомство со архитектурой REST. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.

В данной работе разработаем RESTful веб-API приложение с форматом вывода JSON.

Примерный ход выполнения работы.

Для тестирования веб-интерфейса нам понадобится любой REST-клиент. Удобнее всего пользоваться либо встроенным клиентов в вашей IDE, либо расширением для браузера. В данной лабораторной работе использовано приложение Advanced REST client для браузера Google Chrome. Через менеджер расширений устанавливаем его, ссылка на сайт разработчиков: https://advancedrestclient.com/. Создадим исходную базу данных - электронную библиотеку:

```
CREATE TABLE IF NOT EXISTS `book` (
`id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `authors` varchar(256) DEFAULT NULL,
  `year` year(4) DEFAULT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
INSERT INTO `book` (`id`, `name`, `authors`, `year`) VALUES
(1, 'Язык программирования C', 'Б. Керниган, Д. Ритчи', 2017),
```

```
    (2, 'Алгоритмы и структуры данных', 'Н. Вирт',
2014),
    (3, 'Искусство программирования. Том 3.
Сортировка и поиск', 'Д. Кнут', NULL),
    (4, 'Современные операционные системы', 'Э.
Таненбаум, Х. Бос', 2017),
    (5, 'Ради удовольствия. Рассказ нечаянного
революционера', 'Л. Торвальдс, Д. Даймонд', 2002),
    (6, 'PHP. Объекты, шаблоны и методики
программирования', 'М. Зандстра', 2016),
    (7, 'Getting Started with React', NULL, 2016);
    ALTER TABLE `book` ADD PRIMARY KEY (`id`);
    ALTER TABLE `book` MODIFY `id` int(11) NOT NULL
AUTO_INCREMENT;
```

Теперь создадим обработчики методов, их будет 4:
GET - просмотр всей библиотеки или одной книги
PUT - добавление книги в коллекцию
POST или PATCH - редактирование данных книги
DELETE - удаление книги
Ниже представлен код контроллера (src/controllers.php):

```
    $app->get('/', function () use ($app) {
        return $app->json([
            'Available methods for book' => [
                'GET' => 'All list or one by ID',
                'PUT' => 'Create new book',
                'POST and PATCH' => 'Edit book by
ID',
                'DELETE' => 'Delete book by ID',
            ],
        ]);
    })
    ->bind('homepage')
    ;
    $app->get('book/{id}', function ($id) use ($app)
{
        $sql = 'SELECT * FROM book';
        if ($id) {
            $sql .= ' WHERE id = ?';
        }
```

```
        $result = $app['db']->fetchAll($sql, [$id]);

        return $app->json($result);
    })
    ->convert('id', function ($id) { return (int)
$id; }) # кастуем к числу
    ->value('id', 0) # для вывода всего списка
(значение по умолч.)
    ;
    $app->put('book', function (Request $request) use
($app) {
        if (!$request->get('name')) {
            $app->abort(403, 'Field name is
required');
        }

        $sql = 'INSERT INTO book (`name`, authors,
`year`) VALUES (?, ?, ?)';
        $result = $app['db']->executeUpdate($sql, [
            $request->get('name'),
            $request->get('authors'),
            (int) $request->get('year'),
        ]);
        if (!$result) {
            $app->error(function () { return new
Response('Ошибка БД'); });
        }
        return $app->json(['id' => $app['db']-
>lastInsertId('id')]);
    })
    ;
    $app->match('book/{id}', function (Request
$request, $id) use ($app) {
        $sql = 'SELECT * FROM book WHERE id = ?';
        $book = $app['db']->fetchAssoc($sql, [$id]);
        if (empty($book)) {
            $app->abort(404, "Book {$id} does not
exist");
        }
        $sqlUpdate = 'UPDATE book SET name = ?,
authors = ?, year = ? WHERE id = ?';
        $result = $app['db']-
>executeUpdate($sqlUpdate, [
```

```
            $request->get('name', $book['name']),
            $request->get('authors',
$book['authors']),
            $request->get('year', $book['year']),
            $id,
            ]
        );
        if (!$result) {
            $app->error(function () { return new
Response('Ошибка обновления БД'); });
        }
        return $app->json($app['db']-
>fetchAssoc($sql, [$id]));
    })
    ->method('POST|PATCH') # разрешаем оба типа
    ->assert('id', '\d+') # принимаем только цифры
    ->convert('id', function ($id) { return (int)
$id; }) # кастуем к числу
    ;
    $app->delete('book/{id}', function ($id) use
($app) {
        $sql = 'DELETE FROM book WHERE id = ?';
        $result = $app['db']->executeUpdate($sql,
[$id]);
        if (!$result) {
            $app->error(function () { return new
Response('Ошибка БД'); });
        }
        return $app->json(['id' => $id]);
    })
    ->assert('id', '\d+') # принимаем только цифры
    ->convert('id', function ($id) { return (int)
$id; }) # кастуем к числу
    ;
```

Начинаем проводить тесты.

Главная страница - вывод доступных методов (рис. 8.1):

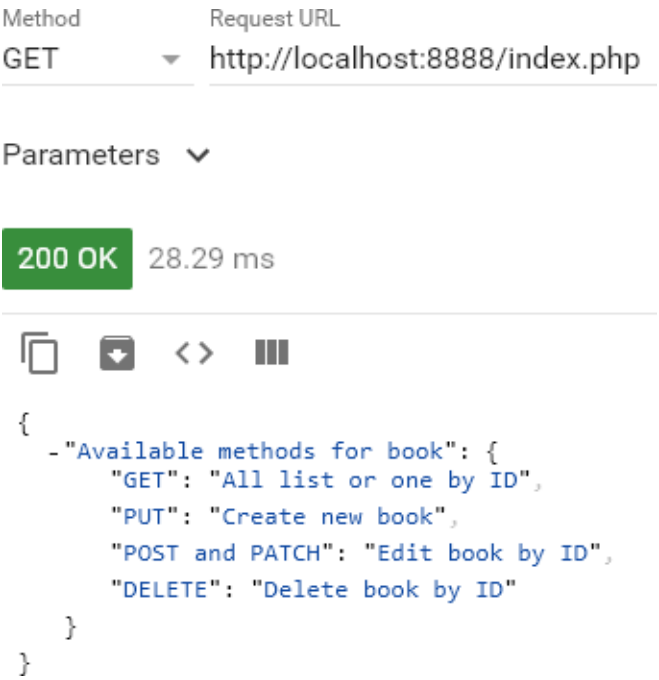

Рис. 8.1. Доступные методы

С помощью метода get можем получить все коллекции библиотеки (рис. 8.2):

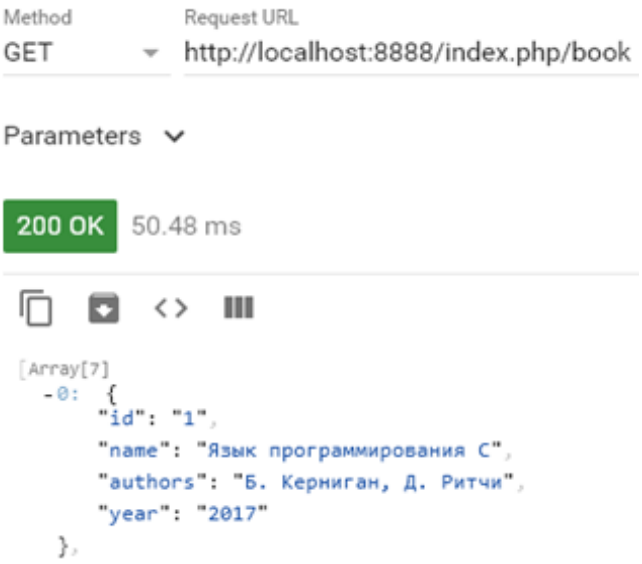

Рис. 8.2. Коллекции библиотеки

Далее получим информацию по конкретной книге (рис. 8.4):

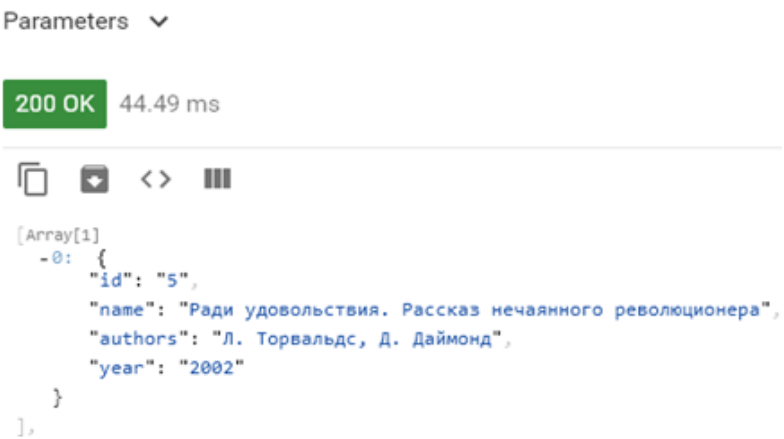

Рис. 8.4. Информация по конкретной книге

Для добавления новой книги воспользуемся методом put (рис 8.5):

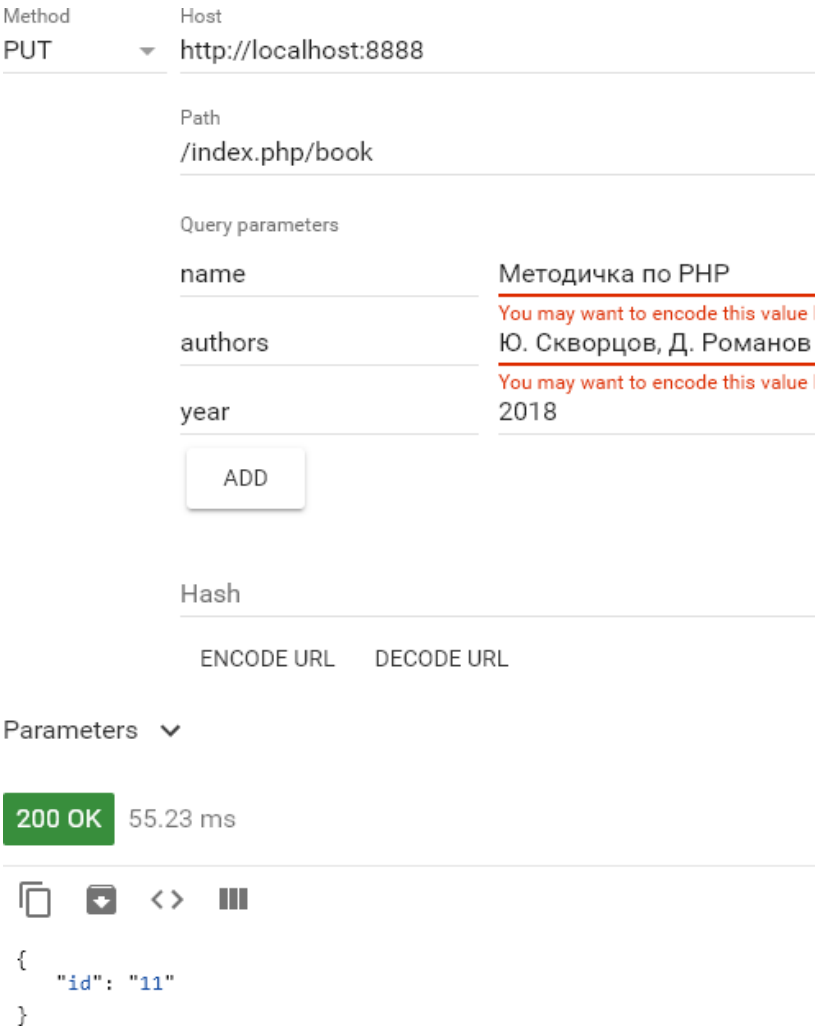

Рис. 8.5. Добавление книги

Для редактирования информации по книге используем метод post (рис. 8.6):

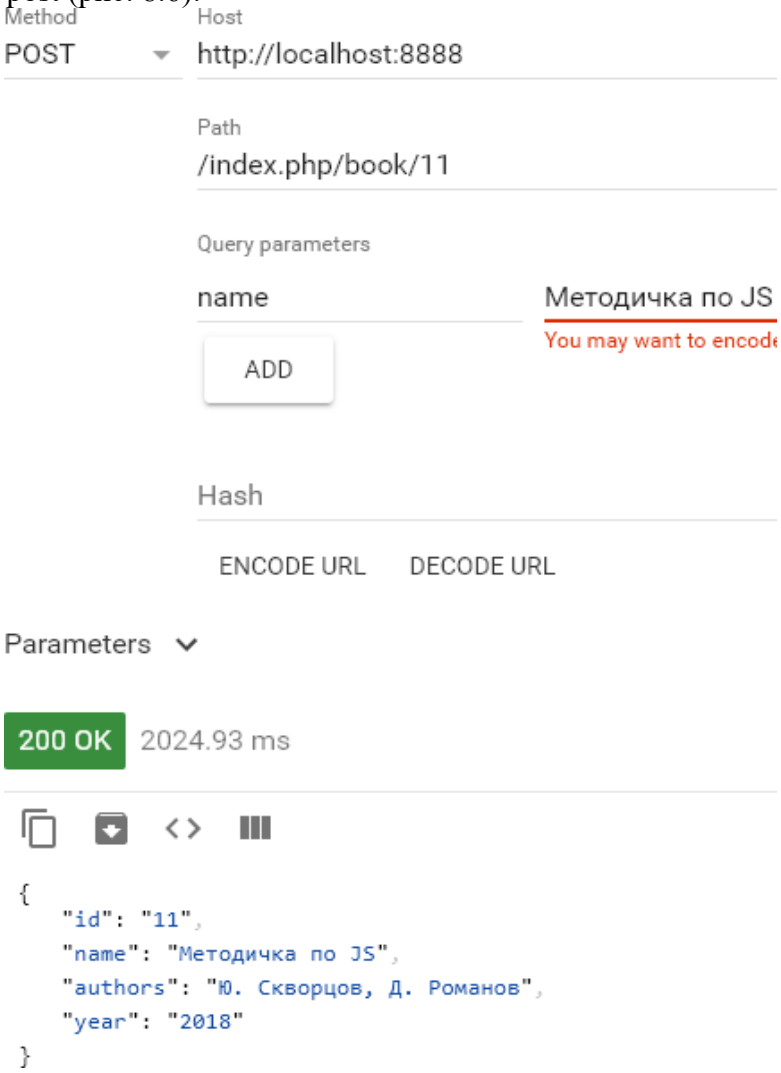

Рис. 8.6. Информация по конкретной книге

Для удаления книги необходимо воспользоваться методом delete.

В ходе данной лабораторной работы создали RESTful веб-API и протестировали его через веб-клиент.

8.5. Контрольные задания

8.5.1. Варианты заданий для реализации

1. Аутентификация пользователя
2. Идентификация пользователя
3. Авторизация пользователя
4. Система учета абитуриентов
5. Сайт автошколы

8.5.2. Вопросы для самопроверки

1. Что такое SOA?
2. Системы мгновенного обмена сообщений.
3. Системы очередей.
4. RabbitMQ.
5. AMQP.
6. Gearman.
7. Websphere.
8. Возможности в Websphere.

ЗАКЛЮЧЕНИЕ

Разработка серверных веб-приложений в наше время связано с обширным комплексом задач, каждая из которых обладает своими особенностями и требует специального подхода. В этом одна из причин сложности изучения процесса проектирования и разработки программного обеспечения в целом. Однако, само функционирование и развитие разрабатываемых ИС напрямую связано с качеством решения этих задач – практика показывает, что любой пользователь работает только с качественными информационными системами, нацеленными на решение пользовательских задач. В выявлении действительных пользовательских потребностей, а также в организации процесса разработки информационных систем, способных их удовлетворить и заключается основная задача построения ИС.

В пособии была сделана попытка вкратце затронуть большинство основных вопросов, касающихся области разработки веб-приложений на языке PHP с использованием популярных фреймворков, а также дать основную информацию, которая может быть полезна при решении возникающих задач. Разумеется, формат пособия не позволяет детально разобрать даже эти вопросы, а ведь в реальности существует еще масса нужных и полезных областей знаний в области разработки веб-приложений, оставшихся за рамками курса. Тем не менее, предоставленный материал, как считают авторы, вполне способен, опираясь на знания, полученные в рамках бакалавриата и систематизируя их, дать минимально необходимый запас для начала работы в такой интересной сфере деятельности, как разработка серверных веб-приложений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Котеров Д. PHP. В подлиннике / Д. Котеров, А. Костарев — Спб.: БХВ-Петербург, 2005. — 1120 с.

2. Костарев А.Ф. PHP 5 / А.Ф. Костарев — М.: БХВ-Петербург, 2008. — 940 с.

3. Зандстра М. PHP: объекты, шаблоны и методики программирования / М. Зандстра. - 3-е издание — М.: Вильямс, 2010. — 560 с.

4. Васвани В. MySQL: использование и администрирование / В. Васвани. — М.: Питер, 2011. — 368 с.

5. Суэринг С. PHP и MySQL. Библия программиста. / С. Суэринг, Т. Конверс, Д. Парк. - 2-е издание. — М.: Диалектика, 2010. — 912 с.

6. Шелдон Р. MySQL 5: базовый курс / Р. Шелдон, Д. Мойе. — М.: Диалектика, 2007. — 880 с.

7. Кузнецов М. MySQL на примерах / М. Кузнецов, И. Симдянов. — Спб.: БХВ-Петербург, 2008. — 952 с.

8. Кузнецов С.Д. Основы баз данных / С.Д. Кузнецов. — 2-е изд. — М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с.

9. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. Коннолли, К. Бегг. — 3-е изд. — М.: Вильямс, 2003. — 1436 с.

10. Гарсиа-Молина Г. Системы баз данных. Полный курс / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. — М.: Вильямс, 2003. — 1088 с.

11. Дари К. PHP и MySQL: создание интернет-магазина / К. Дари, Э. Баланеску. — М.: «Вильямс», 2010. — ISBN 978-5-8459-1602-0.

12. Ленгсторф Д. PHP и jQuery для профессионалов / Д. Ленгсторф. — — М.: «Вильямс», 2010. — 352 с. — ISBN 978-5-8459-1693-8.

13. Суэринг С. PHP и MySQL / С. Суэринг, Т. Конверс. 2-е издание. — М.: «Диалектика», 2010. — 912 с. ISBN 978-5-8459-1640-2.

14. Зервас К. Web 2.0: создание приложений на PHP / К. Зервас. — М.: «Вильямс», 2009. — 544 с. ISBN 978-5-8459-1590-0.

15. Кузнецов М. Объектно-ориентированное программирование на PHP / М. Кузнецов, И. Симдянов. — СПб.: «БХВ-Петербург», 2009. — СПб.: «БХВ-Петербург», 2007. — 608 с. — ISBN 978-5-9775-0142-2.

16. Кузнецов М. PHP 5/6 / М. Кузнецов, И. Симдянов. — СПб.: «БХВ-Петербург», 2009. — 1024 с. — ISBN 978-5-9775-0304-4.

17. Леки-Томпсон Э. PHP 5 для профессионалов / Э. Томпсон, С. Новицки. — М.: «Диалектика», 2006. — 608 с. ISBN 0-7645-7282-2.

18. Кузнецов М. PHP. Практика создания Web-сайтов / М. Кузнецов, И. Симдянов. — СПб.: «БХВ-Петербург», 2008. — 1264 с. — ISBN 978-5-9775-0203-0.

19. Хокинс С. Администрирование веб-сервера Apache и руководство по электронной коммерции / С. Хокинс. - М.: Вильямс, 2001. — 336 с. ISBN 0-13-089873-2.

20. Бэнкер К. MongoDB в действии / К. Бэнкер. - ДМК Пресс, 2014. — 394 с. — ISBN 978-5-97060-057-3.

21. Маклафлин Б. Изучаем Ajax / Б. Маклафлин.— СПб.: «БХВ-Петербург», 2009. . — СПб.: Питер, 2007. — ISBN 978-5-91180-322-3.

22. Хольцнер С. Ajax Библия программиста / С. Хольцнер.— М.: Диалектика, 2009. — 553 с. — ISBN 978-5-8459-1502-3.

23. Крейн Д. Ajax на практике / Д. Крейн, Б. Бибо. — М.: Вильямс, 2007. — ISBN 978-5-8459-1327-2.

24. Вулстон Д. Ajax и платформа .NET 2.0 для профессионалов / Д. Вулстон. — М.: Вильямс, 2007. — 464 с. — ISBN 1-59059-670-6.

25. Крейн Д. AJAX в действии / Д. Крейн, Э. Паскарелло — М.: Вильямс, 2006. — С. 640. — ISBN 1-932394-61-3.

26. Хантер Д. Работа с XML, 4-е издание / Д. Хантер, Д. Рафтер — М.: «Диалектика», 2009. — 1344 с. — ISBN 978-5-8459-1533-7.

27. Хантер Д. XML. Базовый курс / Д. Хантер, Д. Рафтер — М.: Вильямс, 2009. — 1344 с. — ISBN 978-5-8459-1533-7.

28. Тейбор Р. Реализация XML Web-служб на платформе Microsoft .NET / Р. Тейбор. . — М.: Вильямс, 2002. — 464 с. — ISBN 0-672-32088-6.

29. Рамел Д. Joomla! для профессионалов / Д. Рамел. . — 448 с. — ISBN 978-5-8459-1891-8.

30. Колисниченко Д. Joomla! 3.0. Руководство пользователя / Д. Колисниченко. — М.: «Диалектика», 2013. — 256 с. — ISBN 978-5-8459-1864-2.

31. Декстер М. Joomla!: программирование / М. Декстер. — М.: «Вильямс», 2013. — 592 с. — ISBN 978-5-8459-1798-0.

32. Томлинсон Т. CMS Drupal 7: руководство по разработке системы управления веб-сайтом, 3-е издание / Т. Томлинсон. — М.: «Вильямс», 2011. — 560 с. — ISBN 978-5-8459-1743-0.

33. Мелансон Б. Профессиональная разработка сайтов на Drupal 7 / Мелансон Б., Нордин Д. — СПб.: «Питер», 2013. — 688 с. — ISBN 978-5-4461-0054-5.

34. Черных А. Drupal 7 / А. Черных. — «Эксмо», 2011. — 208 с. — ISBN 978-5-699-47059-4.

35. Ромашов В. CMS Drupal: система управления содержимым сайта / В. Ромашов. — СПб.: «Питер», 2010. — 256 с. — ISBN 978-5-49807-241-8.

.

ОГЛАВЛЕНИЕ

Учебное издание

Сапегин Сергей Владимирович
Скворцов Юрий Сергеевич
Романов Дмитрий Валерьевич

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СЕРВЕРНЫХ WEB (PHP 7, FRAMEWORKS)

В авторской редакции

Подписано в печать 22.11.2017.
Формат 60x84/16. Бумага для множительных аппаратов.
Усл. печ. л. 11,5. Уч.-изд. л. 9,6. Тираж 30 экз.
Зак. №

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14

Участок оперативной полиграфии издательства ВГТУ
394026 Воронеж, Московский просп., 14

141