PHP для веба

Изучение PHP без фреймворков

Маттиас Нобак

PHP для веба

Изучение PHP без использования фреймворков

Matthias Noback и Alexey Pyltsyn

Эта книга предназначена для продажи на http://leanpub.com/php-for-web-ru

Эта версия была опубликована 2021-03-08

Оглавление

Введение . . . . . **i**
Для кого эта книга? ii
Подготовка к работе ii
Bash iii
Среда выполнения PHP iv
IDE для PHP iv
Firefox iv
Краткий обзор книги v
Исходный код v
Благодарности vi
Отзывы и предложения vi
История обновлений книги vi
31 января 2021 (перевод от 9 марта 2021) vi

1. Обработка ресурсов **1**
Обработка файла index.html через встроенный веб-сервер 1
Добавление фавиконки 3
Предупреждение по безопасности: корневая директория проекта не должна быть корнем документа 3
Схема работы браузера и сервера 3
Резюме 3
Тесты 3

2. Обработка PHP-скриптов **4**
Ответ: состояние, заголовки и тело 6
Создание ответа 6
Перелинковка с другими страницами 6

Передача значений между запросами 6
Предупреждение по безопасности: не доверяйте пользовательским данным 6
Резюме 6
Тесты 6

3. Формы 7
Отправка данных формы через параметры запроса 7
Предупреждение по безопасности: всегда экранируйте выходные данные 9
Добавление выпадающего списка в форму 9
Отправка данных через тело запроса 9
Резюме 9
Тесты 9

4. Куки 10
Создание куки 12
Использование куки 12
Set-Cookie — заголовок ответа, а Cookie — заголовок запроса 12
Перенаправление после обработки запроса `POST` 12
Предупреждение по безопасности: куки могут быть изменены без вашего ведома 12
Резюме 12
Тесты 12
Упражнение 12

5. Сессии 13
Файлы сессий и сериализованные данные 15
Флеш-сообщения 15
Повсеместное использование флеш-сообщений 15
Резюме 15
Тесты 15

6. Аутентификация 16
Секретная страница 17
Настройка формы входа 17
Проверка имени пользователя и пароля 17

Завершение сессии 17
Резюме 17
Тесты 17

7. Структура проекта 18
HTML-сниппеты для шапки и футера 18
Передача переменных в сниппеты 20
Улучшение флеш-сообщений 20
Начальная загрузка 20
Переход от .html к .php 20
Добавление навигации 20
Стилизация 20
Маршрутизация 20
Резюме 20
Тесты 20
Упражнение 20

8. Создание CRUD. Часть 1 21
Сохранение закодированных данных в файл 22
Добавление тура 22
Валидация формы 22
Вывод отправленных данных в форме 22
Вывод списка туров 22
Резюме 22
Тесты 22
Упражнение 22

9. Создание CRUD. Часть вторая 23
Создание переиспользуемых элементов 24
Редактирование данных тура 24
Удаление туров 24
Резюме 24
Тесты 24
Упражнение 24

10. Загрузка файлов 25
Добавление подробной страницы тура 25

Загрузка файла 27
Обработка загруженных файлов 27
Отображение загруженной картинки 27
Загрузка другой картинки 27
Валидация загруженных файлов 27
Резюме 27
Тесты 27

11. Обработка ошибок 28
Вывод ошибок 28
Использование разных настроек конфигурации на продакшен-сервере 30
Ошибки PHP 30
Резюме 30
Тесты 30

12. Автоматизированное тестирование 31
Установка инструментов тестирования с помощью Composer 31
Первый тест 33
Создание первого браузерного теста 33
Тест для страницы с картинками 33
Начало с чистого листа 33
Решение проблем 33
Резюме 33
Тесты 33
Упражнение 33

13. Заключение 34
Объектно-ориентированное программирование 34
Фреймворки 34
Тестирование 34
Напутственные слова 34

14. Приложение A: Установка PHP на Windows 35

15. Конец отрывка книги 36

Введение

Есть много книг для желающих изучить PHP. Для меня самое серьёзное опасение, связанное с большинством подобных книг, состоит в том, что они охватывают слишком много тем, которые размазываются на сотни страниц, которые нужно как-то проштудировать. Такие книги начинаются с основ программирования на PHP, затем учат создавать сайты с использованием реляционной базы данных, отправлять электронные письма при помощи PHP, разрабатывать консольные приложения и т.д. Это всё полезно и интересно, но, по-моему, это слишком долго. Так я пришёл к мысли, что было бы неплохо написать книгу, которая рассматривает только PHP в контексте веба.

Но сузить тему книги просто-напросто до «PHP для веба» явно недостаточно. Уже написано много всего интересного по созданию веб-приложений на PHP. По большей части эти материалы ориентированы на то, как применять фреймворки вроде Symfony или Laravel, чтобы разрабатывать веб-приложения. Однако я считаю, что важно знать обо всех деталях, из которых состоит сам фреймворк. Когда вы знаете, как работает инструмент изнутри, вы будете лучше справляться с рабочими задачами. Поэтому в этой книге мы не будем использовать фреймворк.

Одновременно с этим стоит заметить, что без применения фреймворка ваше приложение никогда не будет таким качественным и эффективным, как могло быть с ним. Вы столкнётесь с задачами, которые фреймворки решили задолго до вас. А ещё вы, наверняка, создадите проблемы с безопасностью, от которых можно было бы защититься при помощи фреймворка. Поэтому любой код в этой книге следует использовать исключительно в учебных целях.

В последней главе мы обсудим, каким должен быть код в реальном приложении, а также как можно научиться писать такой код.

Для кого эта книга?

Эта книга адресована преимущественно начинающим PHP-разработчикам, которые хотят изучить все стороны применения PHP для создания динамических сайтов. Предполагается, что вы знакомы с основами программирования на PHP, в частности:

- Переменные и присваивание им значений
- Управляющие конструкции, такие как `if`, `else`, `foreach`, `for`
- Строки и конкатенация строк (с помощью `.`)
- Целые числа
- Выражения и сравнения (`<`, `>` и т.д.)
- Ассоциативные массивы (массивы со строковыми ключами и значением) и индексированные массивы (массивы, в которых указываются только значения)

Кроме этого, мне кажется, что данная книга будет полезна разработчикам на других языках программирования, которые хотят изучить PHP, особенно в плане того, как он работает с сетью (по сравнению с Java, Python и т.д.).

Помимо предварительных знаний PHP, я полагаю, что читатель обладает базовыми навыками работы с HTML. Мы будем использовать всего несколько стандартных элементов, но если вы совсем не знаете HTML, я рекомендую прочитать Введение в HTML[1] и Руководство по HTML-формам[2] на портале MDN.

Подготовка к работе

Прежде чем переходить к основному содержанию книги, сначала разберёмся, что нужно сделать перед изучением PHP для веба. Пробежимся по программам, которые обязательно должны быть установлены на компьютере. Я также перечислю список тем книги, с которыми вы, как я думаю, немного знакомы.

[1] https://developer.mozilla.org/ru/docs/Learn/HTML/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2_HTML

[2] https://developer.mozilla.org/ru/docs/Learn/HTML/Forms

Bash

Первое, без чего невозможно обойтись — это Bash. Мы будем использовать его для работы с командной строкой. В Linux и Mac он уже установлен. Чтобы использовать Bash, откройте приложение *Terminal*.

На Windows я рекомендую установить Git[3], уже включающий Git Bash. После установки откройте приложение *Git Bash*.

В окне Bash вы увидите мигающий курсор. Введите следующую команду и нажмите *Enter*:

```
bash --version
```

Отобразится информация об установленной на компьютере версии Bash. Курсор в виде мерцающего прямоугольника говорит о том, что можно ввести другую команду.

Несколько полезных советов по работе с Bash

Комбинация клавиш для копирования и вставки (`Ctrl + C`, `Ctrl + V`) чаще всего не работает в Bash. Если вы хотите скопировать или вставить что-то в *терминал*, щёлкните правой кнопкой мыши в окне *терминала*, чтобы отобразить контекстное меню. В этом меню есть такие кнопки как *Copy* и *Paste*. Как правило, поддерживаются альтернативные сочетания клавиш `Ctrl + Shift + C` и `Ctrl + Shift + V` для копирования и вставки соответственно.

Если нажмёте кнопку `Up`, автоматически появится предыдущая выполненная команда. Вы можете нажать несколько раз кнопку-стрелку `Up`, чтобы переместиться назад по истории.

В дальнейшем, когда в книге будет написано что-то вроде «Выполнить» или «Наберите в командной строке», то я подразумеваю в виду следующее: откройте терминал или Git Bash и выполните в нём команду (по аналогии с выполненной вами ранее команды `bash --version`).

[3] https://git-scm.com/downloads

Среда выполнения PHP

Вам, конечно, потребуется установленная среда выполнения PHP-кода на вашем компьютере. **Примечание: вам не нужен установленный веб-сервер, например, Apache, Nginx или IIS. Мы будем использовать только PHP из командной строки.**

На Linux вы можете установить её с помощью `apt` или любого другого менеджера пакетов. На Mac можно использовать что-то вроде `brew` для установки среды. Для Windows следуйте инструкциям в приложении A.

Чтобы убедиться в правильности установки и заодно проверить работу PHP из командной строки, выполните следующую команду:

```
php -v
```

Вывод команды покажет информацию о текущей версии PHP.

IDE для PHP

Я не буду призывать вас покупать и устанавливать PhpStorm[4], хотя, по моему мнению, это лучший инструмент для написания PHP-кода. Если вы привыкли к другой IDE, которая хорошо интегрирована с PHP-кодом, то смело продолжайте использовать её. Не лишним будет назвать признаки хорошей IDE:

- Она должна указывать на ошибки в коде
- В ней должно поддерживаться форматирование кода
- Она должна предлагать подсказки во время набора кода («автодополнение кода»)

Firefox

И последнее, что нам нужно, — это браузер Firefox[5]. Конечно, я не настаиваю на том, что именно он должен быть вашим браузером по умолчанию. Тем не менее, будет проще, если мы оба будем пользоваться одинаковым браузером.

[4] https://www.jetbrains.com/phpstorm/
[5] https://www.mozilla.org/en-US/exp/firefox/new/

Краткий обзор книги

Перед тем, как перейти к делу, я кратко опишу содержание книги:

В главе 1 показывается, как использовать встроенный PHP-сервер для отдачи (загрузки) *статических* ресурсов: HTML-файлов, изображений, файлов CSS и т.д. В главе 2 мы напишем первый PHP-скрипт. Используя PHP можно привнести динамики в страницы. Глава 3 демонстрирует применение HTML-форм и то, как отправленные через неё данные могут быть обработаны PHP-скриптом. Мы обсудим разницу между запросами `GET` и `POST`. В главе 4 рассказывается про куки, а также как их можно использовать для хранения различных данных в браузере и передавать их последующим запросам. В главе 5 мы познакомимся с *сессионными* куки, чтобы сохранять данные не в браузере, а на сервере. Сочетая изученные приёмы и методы из предыдущих глав, создадим систему для аутентификации в главе 6. Оказавшись в главе 7 займёмся небольшим рефакторингом проекта, чтобы облегчить работу с нашим проектом в оставшихся главах. Затем мы напишем CRUD-страницы, с помощью которых можно будет создавать, обновлять и удалять туры, которые также будут отображаться на сайте. Эта тема поделена на две главы: 8 и 9. Для большей наглядности наших туров добавим возможность загружать картинку для каждого тура в главе 10. В главе 11 разработаем собственное решение для перехвата и обработки ошибок и исключений PHP и покажем их на отдельной странице. В главе 12 объясняется, как протестировать реализованную функциональность с помощью автоматизированного инструмента по выполнению тестов. Итак, мы усвоили все основы и можем переходить к следующей книге или курсу. В заключительной 13 главе я дам рекомендации для дальнейшего изучения.

Исходный код

Все примеры кода из этой книги можно найти в репозитории на GitHub[6]. В конце каждой главы я фиксирую текущее состояние проекта в репозитории системы контроля версий. Вы можете посмотреть код в браузере либо скло-

[6] https://github.com/matthiasnoback/php-for-the-web

нировать проект на свой компьютер (необязательно) с помощью следующей команды:

```
git clone git@github.com:matthiasnoback/php-for-the-web.git
```

Посмотреть изменения в проекте, сделанные в определённой главе, можно с помощью списка коммитов[7]. Достаточно скопировать соответствующий хеш коммита (например, `b596da2`) из списка и выполнить команду:

```
git checkout b596da21d8af0afe91f549efa2eb98da203eeeaa
```

Благодарности

Отзывы и предложения

История обновлений книги

31 января 2021 (перевод от 9 марта 2021)

[7] https://github.com/matthiasnoback/php-for-the-web/commits/master

1. Обработка ресурсов

Обработка файла index.html через встроенный веб-сервер

Самый простой способ создания веб-приложения — “отдача” или обработка файлов с расширением `.html`. В этой главе мы сделаем это с помощью встроенного веб-сервера PHP.

Сначала нужно создать директорию проекта. Она может находится в любом месте на вашем компьютере. Например, я создал директорию проекта по пути `/home/matthias/Projects/php-for-the-web`. Откройте эту директорию в вашей IDE (желательно PhpStorm). Теперь создайте файл `index.html` в директории проекта. Скопируйте следующий HTML-код в этот файл:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Index</title>
    </head>
    <body>
        <h1>This is the index</h1>
    </body>
</html>
```

В терминале перейдите в директорию проекта с помощью `cd` (на Windows это можно сделать, щёлкнув правой кнопкой мыши по директории и выбрать пункт «Open Git-Bash here» в контекстном меню).

```
cd /home/matthias/Projects/php-for-the-web
```

Теперь можно запустить встроенный веб-сервер[1]:

[1] https://www.php.net/manual/ru/features.commandline.webserver.php

```
php -S localhost:8000
```

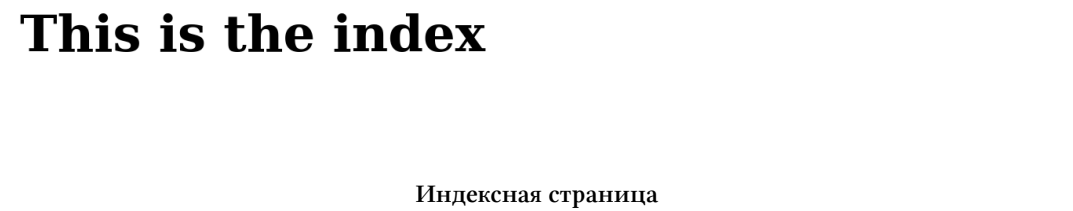
В терминале вы увидите что-то подобное:

```
[...] PHP [...] Development Server (http://localhost:8000) started
```

Итак, PHP-сервер разработки (или также известный как встроенный сервер) запущен. Отлично! Сервер работает по URL-адресу `localhost:8000`. Этот адрес мы указали PHP через опцию командной строки `-S`:

```
php -S localhost:8000
```

Чтобы убедиться, что сервер действительно функционирует, откройте браузер и перейти по URL-адресу http://localhost:8000. Загрузится страница с надписью «This is the index»:



Индексная страница

Добавление фавиконки

Предупреждение по безопасности: корневая директория проекта не должна быть корнем документа

Схема работы браузера и сервера

Резюме

Тесты

2. Обработка PHP-скриптов

В первой главе мы рассмотрели обработку раздачи статических ресурсов, а именно только файлов `index.html` и `favicon.ico`. Напомню: браузер отображает HTML-страницу и загружает `favicon.ico` для отображения фавиконки в браузерной вкладке сайта. Однако, ограничиваясь только возвратом браузеру файлов с расширением `.html` и изображений, мы никогда не создадим настоящее веб-приложение. Файл с `.html` не может обрабатывать данные, отправленные пользователем через форму. Файл с `.html` не может работать с базой данных и отобразить список товаров, полученных из неё. Файл с `.html` не может запомнить моё имя, чтобы поприветствовать меня в духе «Привет, Матиас, с возвращением!». Всё это потому, что файл, оканчивающийся на `.html` — это *статический* ресурс. Когда браузер загружает его во второй раз, это будет всё такой же файл, что и при первом получении.

В этой главе мы собираемся создать *динамические* ресурсы. При повторной загрузке одного и того же динамического ресурса ответ будет отличаться.

Создадим динамический ресурс как новый PHP-файл под названием `random.php`, который поместим в директорию `public/`. Скопируйте и вставьте следующий фрагмент кода в `random.php`:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Your lucky number</title>
    </head>
    <body>
        <h1>Your lucky number is: <?php echo random_int(1, 10); ?></h1>
    </body>
</html>
```

Теперь запустим новый скрипт `random.php`. Так как мы создали файл `random.php` внутри корня документа (`public/`), то его можно открыть из браузера. Перей-

дите по адресу http://localhost:8000/random.php и вы увидите что-то похожее на это:

Your lucky number is: 7

Ваше счастливое число — 7

Скорее всего, вам выпадет другое число, потому что мы воспользовались функцией `random_int()`[1] для генерации случайного числа. Обновите страницу, чтобы убедиться, что было сгенерировано новое число.

Кстати, если при попытке открыть страницу в вашем браузере показывается надпись «Unable to connect», то значит PHP-сервер не был запущен. В таком случае откройте терминал, перейдите в директорию проекта и выполните команду:

```
php -S localhost:8000 -t public/
```

[1] https://www.php.net/random_int

Ответ: состояние, заголовки и тело

Создание ответа

Перелинковка с другими страницами

Передача значений между запросами

Предупреждение по безопасности: не доверяйте пользовательским данным

Резюме

Тесты

3. Формы

Во второй главе мы узнали про способ обработки PHP-скриптов встроенным сервером PHP. Мы также увидели, как при помощи параметров запроса (являющихся частью URL-адреса) можно передавать произвольные данные в эти скрипты. Это хороший вариант, когда входные данные поступают из самого приложения (как случайное число в `/random.php`). Но если нужно принимать данные от пользователя, следует воспользоваться *HTML-формой*. Форма гораздо более удобнее и привычнее для пользователя, так как ему не потребуется изменять URL-адрес вручную. Пользователю достаточно ввести информацию в поля формы, выбрать элементы из выпадающего списка, поставить галочку в чекбоксе, а затем отправить эти данные на сервер.

Отправка данных формы через параметры запроса

Разместим простую форму на страницу `/kittens.php`. В этой форме будет только поле для ввода номера. Данные формы отправляются в виде параметров запроса.

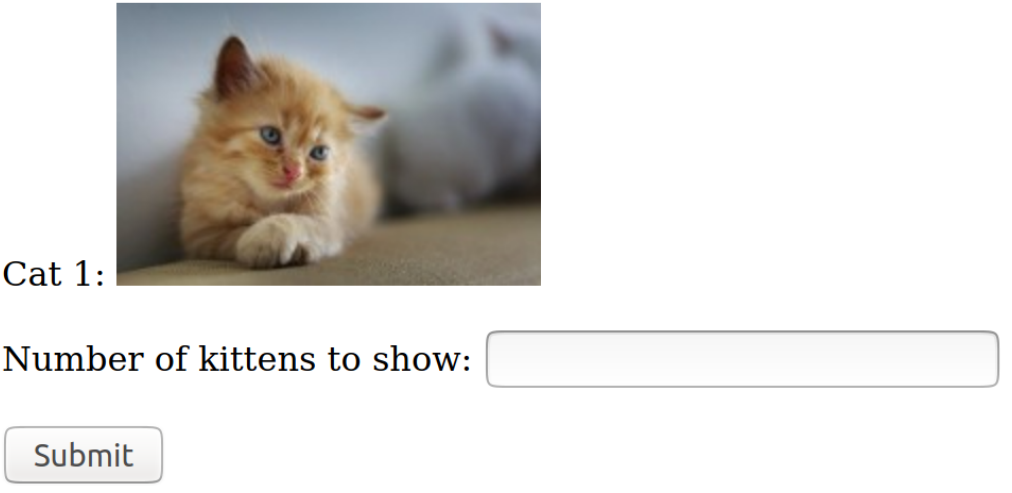
Сначала добавьте форму в `kittens.php` перед закрывающим тегом `</body>`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Kittens</title>
</head>
<body>
<?php

$numberOfKittens = isset($_GET['number']) ? (int) $_GET['number'] : 1;
```

```
// ...
?>
    <form>
        <div>
            <label for="number">
                Number of kittens to show:
            </label>
            <input name="number" id="number">
        </div>
        <div>
            <button type="submit">Submit</button>
        </div>
    </form>
</body>
</html>
```

Теперь перейдите по адресу http://localhost:8000/kittens.php и в снизу страницы должна отобразиться наша форма:



Новая форма на `/kittens.php`

Введите любое число в поле и отправьте форму. После этого вы увидите

указанное вами количество картинок. Здорово!

Но как это работает? Обратите внимание на URL-адрес: если вы ввели 4, то URL-адрес будет http://localhost:8000/kittens.php?number=4. Как раз то, что мы ожидали. Но есть небольшая проблема с юзабилити: после отправки формы поле номера остаётся пустым. Это неудобно, потому что для исправления такой ошибки пользователь вынужден снова ввести указанное число, а не просто изменить уже набранное количество.

Предупреждение по безопасности: всегда экранируйте выходные данные

Добавление выпадающего списка в форму

Отправка данных через тело запроса

Резюме

Тесты

4. Куки

HTTP называют протоколом *без состояния*. Сервер обрабатывает запрос и возвращает ответ, и на этом всё заканчивается: сервер больше ничего не знает про этот запрос, а вместо этого ожидает обработку следующего запроса. Между всеми этими запросами не прослеживается никакой связи, нет даже способа узнать, что новый запрос пришёл от того же человека, что и предыдущий, кроме как только передавать данные от запроса к запросу. Вот как раз для этого и нужны *куки* (cookies).

Давайте сохраним имя пользователя и будем показывать его на каждой странице. Начнём с добавления новой страницы, на которой пользователь может указать своё имя. Создайте скрипт `name.php` в директории `public/` и вставьте следующий код в этот файл:

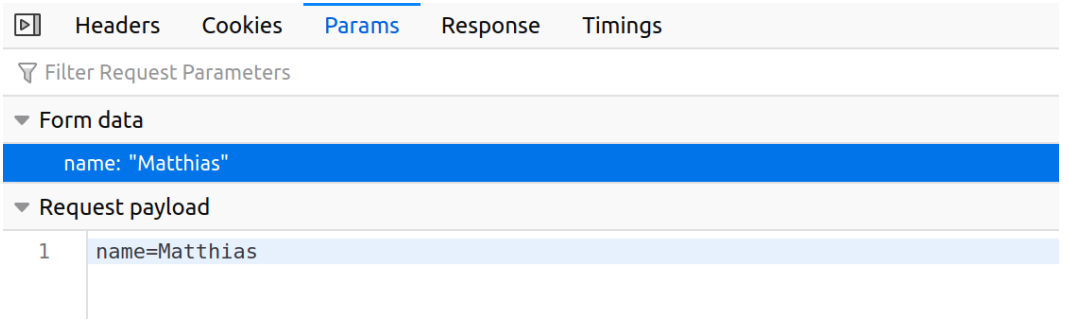
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Name</title>
</head>
<body>
<form method="post">
    <p>
        <label for="name">
            Your name:
        </label>
        <input type="text" name="name" id="name">
    </p>
    <p>
        <button type="submit">Submit</button>
    </p>
</form>
```

```
</body>
</html>
```

Перейдите по адресу http://localhost:8000/name.php, чтобы увидеть новую форму:

Новая форма для ввода имени пользователя

Откройте панель *Network*, введите имя и отправьте форму. В списке запросов должен появится новый запрос типа `POST` для получения `name.php`. Если кликнуть на него и выбрать вкладку *Params*, можно убедиться, что имя было отправлено:



Отправленные данные формы

Создание куки

Использование куки

Set-Cookie — заголовок ответа, а Cookie — заголовок запроса

Перенаправление после обработки запроса `POST`

Предупреждение по безопасности: куки могут быть изменены без вашего ведома

Резюме

Тесты

Упражнение

5. Сессии

В четвёртой главе мы узнали про использование куки для передачи данных между запросами. Мы закончили предостережением, поскольку куки представляет собой HTTP-заголовки, то они могут изменяться без вашего участия, следовательно им нельзя доверять и просто так использовать в коде. Общедоступная видимость содержимого куки — ещё одна проблема с безопасностью. Все эти факторы не позволяют использовать куки для хранения конфиденциальной информации или данных, которые должны изменяться только вами как разработчиком.

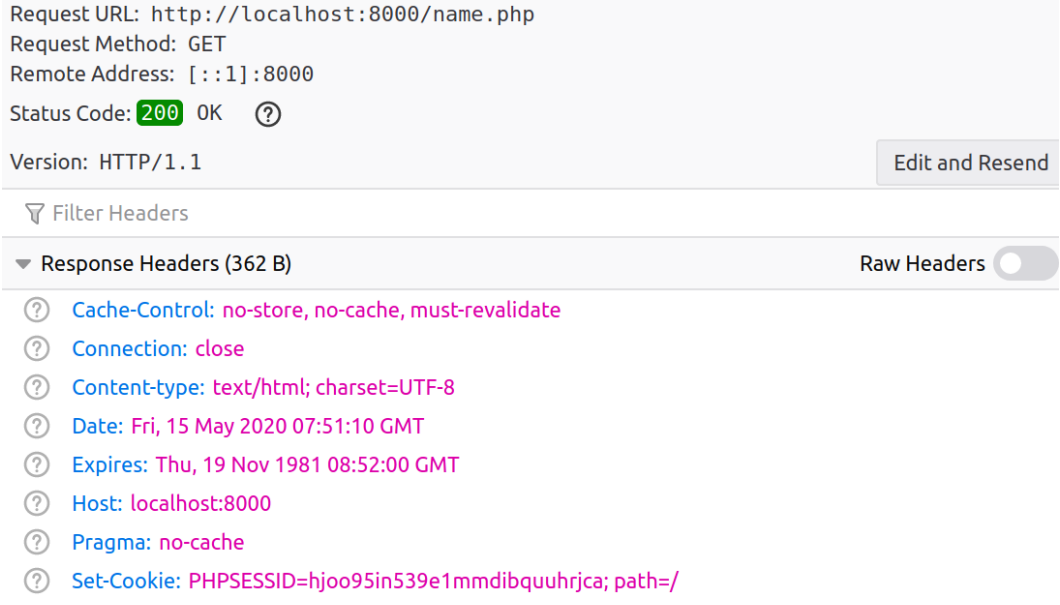
В тех случаях, когда куки не подходят, но всё равно нужно передавать определённую информацию между запросами, можно прибегнуть к помощи *сессий*. За данные в куках отвечает браузер, за данные в сессии — сервер. При таком способе у вас появляется полный контроль над данными, когда никто другой кроме вас не сможет посмотреть или изменить их. Посмотрим, как это работает.

Мы собираемся переписать скрипт `name.php`, чтобы сохранить имя пользователя в сессии, а не в куках. Первое, что нужно сделать в скрипте — это начать сессию с помощью функции [`session_start()`] (https://www.php.net/session_start). Данные сессии будут сохранены в суперглобальную переменной `$_SESSION`, которая является ассоциативным массивом. В этом случае мы присваиваем значение `$_POST['name']` в `$_SESSION['name']`:

```php
<?php
session_start();

if (isset($_POST['name'])) {
    $_SESSION['name'] = $_POST['name'];
    header('Location: /random.php');
    exit;
}
?>
```

Давайте разберём этот код. Но для начала откройте панель *Storage* и удалите все существующие куки, чтобы они нас не сбивали с толку. Теперь откройте панель *Network* и перейдите по URL-адресу http://localhost:8000/name.php. В списке запросов вы увидите запрос `GET`, загружающий `name.php`. Кликните на этот запрос и перейдите в секцию *Response Headers* на вкладке *Headers*:



Заголовки ответов `GET`-запроса по `name.php`

Файлы сессий и сериализованные данные

Флеш-сообщения

Повсеместное использование флеш-сообщений

Резюме

Тесты

6. Аутентификация

Как правило, веб-приложение состоит из:

1. Публичной части
2. Раздела для авторизованных пользователей
3. Панели администратора («админки»)

Публичные страницы доступны для так называемых *анонимных* пользователей (или обычных посетителей сайта). Для просмотра других страниц, таких как списка заказов или профиля, пользователем сначала нужно пройти аутентификацию. Аутентификация — проверка подлинности пользователя. После проверки имени пользователя и пароля мы знаем, что перед нами действительно этот самый пользователь, а не кто-то другой. В конце концов, никто не должен знать чужой пароль, это ведь секрет. Как вы понимаете, в этом предположении скрывается много проблем с безопасностью: пользователь может написать свой пароль на стикере рядом с монитором. Или он может поделиться своими регистрационными данные с внуком. Или у него может быть задан пароль, который легко угадать. Очень много вариантов, и я настоятельно рекомендую вам углубиться в эту тему веб-безопасности, поскольку она выходит за рамки этой книги. Я настоятельно рекомендую OWASP как источник инструкций по безопасности для программистов. Также изучите их шпаргалку по аутентификации[1].

[1] https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

Секретная страница

Настройка формы входа

Проверка имени пользователя и пароля

Завершение сессии

Резюме

Тесты

7. Структура проекта

HTML-сниппеты для шапки и футера

Наверное, в процессе чтения вы обратили внимание, что мы копируем и вставляем один и тот же код в нескольких местах приложения. Прежде всего, мы дублируем большую часть основной HTML-разметки (`<!DOCTYPE html><html...`) на каждой новой странице. Конечно, содержимое внутри `<body>...</body>` будет разное у каждой странице, но вот остальная (базовая) HTML-часть может вставляться из одного общего файла, так называемого сниппета. Давайте начнём с разделения одинакового для всех страниц HTML-кода из `login.php` на повторно используемые сниппеты. Сначала создайте файл `_header.php` в корневой директории проекта (не в публичной директории `public/`, т.к. это логическая часть кода, а не полноценная страница). В этом файле будет храниться общий код для верхней HTML-части (обычно известной как шапка сайта):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login</title>
</head>
<body>
```

Аналогичное сделаем и с нижней HTML-частью, также известной как футер. Создайте файл `_footer.php` и скопируйте туда следующий код:

```
</body>
</html>
```

Теперь вернитесь обратно к `login.php` и подключите оба только что созданных файла в соответствующих местах:

```php
<?php
include(__DIR__ . '/../_header.php');
?>
<form method="post">
    <div>
        <label for="username">
            Username:
        </label>
        <input type="text" name="username" id="username">
    </div>
    <div>
        <label for="password">
            Password:
        </label>
        <input type="password" name="password" id="password">
    </div>
    <div>
        <button type="submit">Submit</button>
    </div>
</form>
<?php
include(__DIR__ . '/../_footer.php');
```

Передача переменных в сниппеты

Улучшение флеш-сообщений

Начальная загрузка

Переход от .html к .php

Добавление навигации

Стилизация

Маршрутизация

Резюме

Тесты

Упражнение

8. Создание CRUD. Часть 1

В этой главе мы применим все рассмотренные ранее техники и создадим CRUD-интерфейс, который есть у большинства приложений.

CRUD расшифровывается как «Create - Read - Updated - Delete». И хотя, поговорив с людьми, вы поймёте, что далеко не все разделяют этот подход, тем не менее он может стать мощным способом для открытия возможностей вашего приложения. В следующих разделах мы собираемся расширить сферу нашего онлайн-бизнеса. Мы станем серьёзным туристическим агентством, которое предлагает туры из Амстердама в различные интересные города поблизости. Показать список всех имеющихся туров — первое, что нам потребуется сделать. Кроме того, мы уверены, что путешествия должны быть доступными для каждого человека, поэтому уделим особое внимание этому.

Перед тем, как показать список туров на сайте, их сначала нужно где-нибудь создать. Конечно, мы можем определить их прямо в коде в виде массива:

```
$tours = [
    [
        'destination' => 'Berlin',
        'number_of_tickets_available' => 10,
        'is_accessible' => true
    ],
    // and so on...
];
```

Но в таком случае нам придётся обновлять на сервере сайт каждый раз, когда потребуется добавить новый тур, отредактировать существующий или удалить его. И вот тут на помощь приходит CRUD, при помощи которого пользователь сможет выполнить всё вышеперечисленное прямо на самом сайте. Авторизованный пользователь с правами администратора будет иметь возможность отредактировать тур, добавить новый или удалить существующий.

В свою очередь обычные пользователи смогут посмотреть список всех туров и перейти на страницу тура, чтобы подробнее узнать о нём. Большинство веб-приложений используют базу данных для хранения такого рода данных, но так как в данной книге не рассматриваются базы данных и не предполагается опыт работы с ними от читателей, то туры будут хранится в ƒайле.

Сохранение закодированных данных в файл

Добавление тура

Валидация формы

Вывод отправленных данных в форме

Вывод списка туров

Резюме

Тесты

Упражнение

9. Создание CRUD. Часть вторая

К данному моменту нам удалось создать форму, которая позволяет пользователю добавить новый тур. Следующий шаг — дать ему возможность отредактировать тур, это пригодится, в случае если пользователь опечатался, или просто желает изменить информацию по туру. Прежде чем приступить к реализации новой функциональности, нужно придумать способ идентификации каждого взятого тура. Поэтому при создании тура и добавлении его в массив `$toursData`, нам также нужно задать для него свойство `id`, служащее его уникальным идентификатором (ID). Уникальный идентификатор (ID) должен быть неиспользованным ранее числом. В нашем случае идентификатор каждого тура будет вычисляться из количества туров в массиве `$toursData`, увеличенное на единицу. Таким образом, если использовать такой алгоритм для генерации ID, то идентификатор первого тура будет 1, у второго — 2 и так далее. При условии, что мы не будем удалять элементы из массива, идентификаторы будут оставаться уникальными. Изменим код скрипта `create-tour.php`, чтобы в нём устанавливался идентификатор каждого тура:

```
if (count($formErrors) === 0) {
    // Provide a unique ID for this new tour:
    $normalizedData['id'] = count($toursData) + 1;

    $toursData[] = $normalizedData;
```

Если вы не удалили файл `tours.json` в конце предыдущей главы, сейчас самое время это сделать. У текущих элементов в `tours.json` нет идентификаторов, поэтому чтобы гарантировать, что у всех туров будет указан ID, нам нужно удалить существующие туры. В качестве альтернативы можно вручную добавить идентификаторы к существующим JSON-данным, но проще удалить файл с ними.

Теперь посмотрим, работает ли код, как мы задумали. Откройте браузер и перейдите на страницу по адресу http://localhost:8000/create-tour, заполните и отправьте форму. Посмотрите на содержимое на `tours.json`: там вы увидите один тур с `"id": 1`:

```
[
    {
        "destination": "Berlin",
        "number_of_tickets_available": 10,
        "is_accessible": true,
        "id": 1
    }
]
```

Создайте несколько других туров, каждый из которых получит собственный идентификатор.

Создание переиспользуемых элементов

Редактирование данных тура

Удаление туров

Резюме

Тесты

Упражнение

10. Загрузка файлов

В каталоге туров явно не хватает красивой фотографий самого места отдыха. Так почему бы реализовать возможность администраторам загрузить фотографию тура, которую отобразить на отдельной подробной странице по туру (которого, кстати, пока у нас нет). Однако следует учесть, что в загрузка пользовательских файлов на сервер представляет собой ещё больший потенциальную угрозу безопасности, чем что-либо, что мы делали ранее. Люди могут загружать некорректны файлы: файлы, которые похожи на картинки, но на деле являются программами, либо изображения с неприемлемым содержанием и т.д. Однако, несмотря на такие возможные сценарии, у администратора должна быть возможность загружать картинки.

Эту задачу можно разбить на следующие шаги:

1. Добавить поддержку загрузки файлов в форму создания и редактирования тура;
2. После успешной валидации формы получить загруженный от пользователя файл и переместить его в постоянное место хранения внутри публичной директории;
3. Сохранить имя файла в файл `tours.json`;
4. Показать картинку на подробной странице тура.

Добавление подробной страницы тура

Для начала займёмся отдельной или подробной страницей тура, на которой покажем все данные о туре, включая, конечно, и новую картинку. Пока у тура есть только направление, количество доступных билетов и опция доступности. Что ж, отобразим эту имеющуюся информацию. Идентификатор тура на подробную страницу будет передаваться в виде параметра запроса. Затем воспользуемся существующей функцией `load_tour_data()`, чтобы загрузить данные тура по идентификатору из строки запроса. В `pages/` создадим новый файл `tour.php` со следующим содержимым:

```
<?php

include(__DIR__ . '/functions/tour-crud.php');

include(__DIR__ . '/../bootstrap.php');

if (!isset($_GET['id'])) {
    header('Location: /list-tours');
    exit;
}

$tourId = (int)$_GET['id'];
$tourData = load_tour_data($tourId);

include(__DIR__ . '/../_header.php');

?>
<h1>Tour to <?php
    echo htmlspecialchars($tourData['destination'], ENT_QUOTES);
?></h1>
<p>This tour is <?php echo $tourData['is_accessible']
    ? 'accessible'
    : 'not accessible'; ?>.</p>
<p>There are <?php
    echo htmlspecialchars(
        $tourData['number_of_tickets_available'], ENT_QUOTES
    );
?> tickets available.</p>
<?php

include(__DIR__ . '/../_footer.php');
```

Как обычно для новой страницы в массив `$urlMap` точки входа `index.php` добавим новый маршрут:

```
$urlMap = [
    '/create-tour' => 'create-tour.php',
    '/list-tours' => 'list-tours.php',
    '/edit-tour' => 'edit-tour.php',
    '/delete-tour' => 'delete-tour.php',
    '/tour' => 'tour.php',
    // ...
];
```

Загрузка файла

Обработка загруженных файлов

Отображение загруженной картинки

Загрузка другой картинки

Валидация загруженных файлов

Резюме

Тесты

11. Обработка ошибок

Как только во второй главе мы начали использовать PHP-сервер для обработки скриптов с расширением `.php`, нужно озадачиться отображением ошибок в браузере. Тогда я упоминал, что следует чётко различать сайт, работающий на вашем компьютере и сайт, запущенный на публичном сервере. Обычно эти две версии сайта называются по-разному. Работающий сайт на вашем компьютере запущен «локально" или на вашем «сервере для разработки». Когда сайт работает на публичном сервере, говорят, что он «развёрнут» на «рабочем или продакшен-сервере». Мы используем разные формулировки в зависимости от контекста и окружений, поскольку сайт по-разному работает «локально» и на «продакшен-сервере». В этой главе мы усовершенствуем обработку ошибок с учётом окружения, в которой работает сайт.

Вывод ошибок

Для начала создадим страницу для отображения ошибки на сайте. В директории `pages/` создадим новый скрипт `oops.php`. Затем добавьте его в массив маршрутов `$urlMap` в `index.php`, чтобы страница была доступен из браузера:

```
$urlMap = [
    '/oops' => 'oops.php',
    // ...
];
```

На самом деле это временная страница, которую мы позже удалим, она будет служить временным местом для просмотра сгенерированных ошибок. Первый тип ошибок, с которым нам нужно разобраться — *исключение*[1]. Исключения полезны в ситуациях, когда нужно указать, что скрипт не может продолжить дальнейшую работу. У нас был такой случай в девятой главе в функции

[1] https://www.php.net/manual/ru/language.exceptions.php

`load_tour_data()`. Функция «выбрасывает» исключение, если были запрошены данные по несуществующему туру:

```
function load_tour_data(int $id): array
{
    $toursData = load_all_tours_data();

    foreach ($toursData as $tourData) {
        if ($tourData['id'] === $id) {
            return $tourData;
        }
    }

    throw new RuntimeException('Could not find tour with ID ' . $id);
}
```

В нашей тестовой странице `oops.php` мы также выбросим исключение, чтобы посмотреть, как оно выглядит для пользователя:

```
<?php

throw new RuntimeException('Something went wrong');
```

Запустите PHP-сервер, если ещё это не было сделано:

```
php -S 0.0.0.0:8000 -t public/ -c php.ini
```

Теперь перейдите по URL-адресу http://localhost:8000/oops, и вы увидите фатальную ошибку:

Fatal error: Uncaught RuntimeException: Something went wrong in /app/pages /oops.php:3 Stack trace: #0 /app/public/index.php(21): include() #1 {main} thrown in **/app/pages/oops.php** on line **3**

Фатальная ошибка: необработанное исключение RuntimeException

Нам показываются ошибка из-за того, что в нашем собственном файле `php.ini` мы активировали PHP-опцию `display_errors`, установив для неё значение `On`. При помощи опции командной строки `-c` мы указали PHP-серверу использовать этот конфигурационный файл.

Подобные сообщения об ошибках крайне полезны в процессе разработки, т.к. они помогают быстрее исправить возникающие проблемы. При этом совершенно неуместно выводить ошибки в браузере вашего посетителя сайта.

Использование разных настроек конфигурации на продакшен-сервере

Ошибки PHP

Резюме

Тесты

12. Автоматизированное тестирование

Я всегда расстраиваюсь, когда авторы книги рассказывают об автоматизации тестирования только в последней главе, и что иронично, сам так поступил! Что ж, давайте обсудим автоматизированное тестирование.

Тестирование — очень важная составляющая в работе разработчика. До сих пор мы вносили изменения в код и «тестировали» их вручную в браузере. Это называется *исследовательским* (ручным) тестированием, которое делает человек. В этой заключительной главе я познакомлю вас с методом тестирования, выполняемого компьютером. Благодаря нему вы сможете сэкономить кучу своего времени, которое было бы потрачено на ручное тестирование.

Но больше всего в автоматизированном тестировании мне нравится, что имеющийся набор тестов можно выполнить в любое время. В процессе разработки изменения в одном файле могут привести к нарушениям в другом файле. Как раз выполнение тестов после каждого изменения выявит такие проблемы. Таким образом тесты становятся своего рода страховочной сетью. По этой причине их ещё называют *регрессионными тестами*, поскольку они помогают вам двигаться вперёд.

Установка инструментов тестирования с помощью Composer

Прежде чем писать и выполнять тесты, сначала нужно установить все необходимые инструменты для этого. Есть несколько отличных библиотек, предназначенных для автоматизированного тестирования сайтов. Большинство PHP-фреймворков уже включают их, а некоторые из них даже предлагают собственные решения. Для тестирования нашего приложения я буду использовать

PHPUnit[1] вместе с Panther[2]. Вы можете установить их только при помощи Composer, инструмента для загрузки PHP-пакетов (например, библиотек и фреймворков) в ваш проект. Перед этим нужно установить сам Composer на компьютер. В *терминале* перейдите в корневую директорию проекта. Затем следуйте инструкциям по установке на официальном сайте Composer[3]. Когда закончите с установкой, в директории проекта появится файл `composer.phar`. Теперь можно перейти к загрузить пакеты PHPUnit и Panther:

```
php composer.phar require --dev \
   phpunit/phpunit symfony/panther symfony/css-selector symfony/mime
```

Этот процесс займёт некоторое время, и завершится примерно таким выводом:

```
Using version ^0.7.1 for symfony/panther
...
./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 15 installs, 0 updates, 0 removals
  - ...
  - Installing symfony/panther (v0.7.1): Downloading (100%)
...
Writing lock file
Generating autoload files
```

В результате установки пакетов создалась новая директория `vendor/` (если вы используете Git, обязательно добавьте `/vendor/` в файл `.gitignore`). В директории `vendor/` находится весь сторонний код (на данный момент это Panther, PHPUnit и их зависимости). Кроме этого, теперь в корневой директории проекта есть файл `composer.json`, который содержит список всех пакетов (зависимостей), которые вы установили. Да, для создания автотестов нам понадобится много пакетов. Но это только потому, что все эти пакеты берут всю сложную работу на себя, облегчая нам жизнь.

[1] https://phpunit.de/
[2] https://github.com/symfony/panther
[3] https://getcomposer.org/download/

Первый тест

Создание первого браузерного теста

Тест для страницы с картинками

Начало с чистого листа

Решение проблем

Резюме

Тесты

Упражнение

13. Заключение

Объектно-ориентированное программирование

Фреймворки

Тестирование

Напутственные слова

14. Приложение A: Установка PHP на Windows

15. Конец отрывка книги

Спасибо, что посмотрели ознакомительный фрагмент книги! Надеюсь, вас так заинтересовала книга, что вы купите полную её версию. По следующей ссылке вы можете купить книгу всего за 9 долларов: http://leanpub.com/learning-php-for-the-web-without-a-framework/c/SAMPLE.

Вы можете связаться со мной в Twitter (@matthiasnoback[1] или отправить мне электронное письмо по адресу info@matthiasnoback.nl[2].

[1] https://twitter.com/matthiasnoback
[2] mailto:info@matthiasnoback.nl