

самоучитель

Максим Кузнецов, Игорь Симдянов

PHP 7

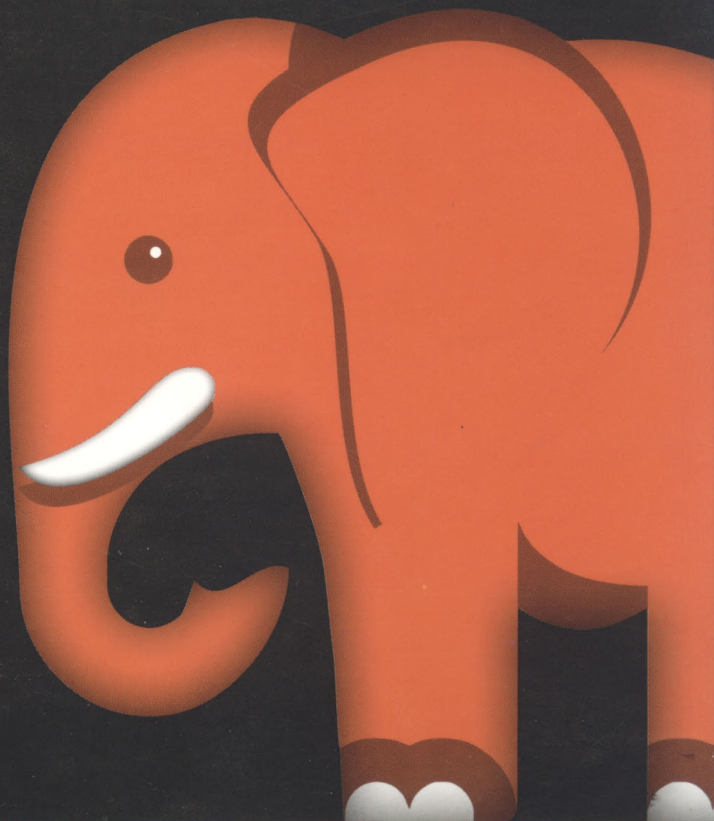
Новинки PHP 7

Шаблоны
проектирования,
итераторы и генераторы

Приемы работы с СУБД
PostgreSQL

Взаимодействие
с базами данных NoSQL
(Redis и подобными)

100 заданий



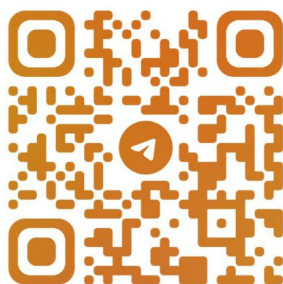
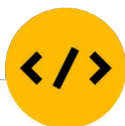
Материалы
на www.bhv.ru



Максим Кузнецов
Игорь Симдянов

самоучитель

PHP 7



@CODELIBRARY_IT

УДК 004.438 PHP
ББК 32.973.26-018.1
К89

Кузнецов, М. В.

К89 Самоучитель PHP 7 / М. В. Кузнецов, И. В. Симдянов. — СПб.: БХВ-Петербург, 2018. — 448 с.: ил. — (Самоучитель)

ISBN 978-5-9775-3817-6

Книга опытных разработчиков описывает последнюю версию языка разработки серверных сценариев PHP 7. Рассмотрены все нововведения языка и связанные с ними изменения в разработке современных Web-сайтов. Изложение ведется с упором на объектно-ориентированное программирование, шаблоны проектирования, итераторы, генераторы, а также взаимодействие с современными базами данных (PostgreSQL и Redis).

В конце глав приведены более 100 заданий для закрепления материала и освоения не вошедших в книгу разделов языка. Электронный архив с исходными кодами доступен на сайтах издательства и GitHub.

Для Web-разработчиков

УДК 004.438 PHP
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Дизайн обложки	<i>Марины Дамбиевой</i>

Подписано в печать 30.03.18.
Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 36,12.
Тираж 1500 экз. Заказ № 6274.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано с готового оригинал-макета
ООО «Принт-М», 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

ISBN 978-5-9775-3817-6

© ООО "БХВ", 2018
© Оформление. ООО "БХВ-Петербург", 2018

Оглавление

Предисловие	11
Объектно-ориентированный подход.....	11
PostgreSQL и Redis	11
Задания	12
Исходные коды	12
Благодарности.....	13
 Глава 1. Что представляет собой PHP?	15
1.1. Достоинства и недостатки.....	15
1.2. Структура PHP	17
1.3. Сопутствующие технологии	17
Задание	18
 Глава 2. Установка PHP	19
2.1. Установка в Windows	19
2.2. Установка в Mac OS X	21
2.3. Установка в Linux (Ubuntu).....	22
2.4. Встроенный сервер.....	22
2.5. Файл hosts.....	23
2.6. Вещание вовне	24
2.7. Настройка PHP.....	24
2.8. Расширения	26
2.9. Документация.....	27
Задания	27
 Глава 3. Быстрый старт.....	29
3.1. Скрипты.....	29
3.2. Начальные и конечные теги.....	31
3.3. Использование точки с запятой.....	32
3.4. Составные выражения. Фигурные скобки	33
3.5. Комментарии.....	34
3.6. Включение PHP-файла.....	36
Задания	37

Глава 4. Переменные и типы данных	39
4.1. Объявление переменной. Оператор =.....	39
4.2. Типы данных.....	40
4.3. Целые числа	41
4.4. Вещественные числа	42
4.5. Логический тип	43
4.6. Строки.....	44
4.7. Кавычки	44
4.8. Оператор <<<.....	47
4.9. Обращение к неинициализированной переменной. Замечания (Notice).....	48
4.10. Специальный тип <i>null</i>	48
4.11. Уничтожение переменной. Конструкция <i>unset()</i>	49
4.12. Проверка существования переменной. Конструкции <i>isset()</i> и <i>empty()</i>	50
4.13. Определение типа переменной.....	52
4.14. Неявное приведение типов	54
4.15. Явное приведение типов	55
4.16. Динамические переменные.....	58
Задания	59
Глава 5. Классы и объекты	61
5.1. Собственные типы данных	61
5.2. Создание класса	62
5.3. Разделение классов и остального кода	63
5.4. Создание объекта.....	65
5.5. Область видимости переменных класса	66
5.6. Спецификаторы доступа	67
5.7. Статические переменные класса	68
5.8. Ссылки на переменные.....	69
5.9. Клонирование объектов	70
Задания	71
Глава 6. Константы.....	73
6.1. Объявление константы. Функция <i>define()</i>	73
6.2. Проверка существования константы.....	75
6.3. Динамическое имя константы	76
6.4. Предопределенные константы.....	77
6.5. Абсолютный и относительный пути к файлу	78
6.6. Константы класса.....	79
Задания	80
Глава 7. Операторы	81
7.1. Объединение строк. Оператор "точка"	81
7.2. Конструкция <i>echo</i> . Оператор "запятая"	82
7.3. Арифметические операторы	83
7.4. Поразрядные операторы	87
7.5. Операторы сравнения.....	92
7.6. Приоритет выполнения операторов	95
Задания	96

Глава 8. Условия	97
8.1. Условный оператор <i>if</i>	97
8.2. Логические операторы	99
8.3. Условный оператор $x ? y : z$	104
8.4. Оператор <i>??</i>	105
8.5. Переключатель <i>switch</i>	105
8.6. Оператор <i>goto</i>	109
Задания	110
Глава 9. Циклы	111
9.1. Цикл <i>while</i>	111
9.2. Цикл <i>do ... while</i>	116
9.3. Цикл <i>for</i>	117
Задания	121
Глава 10. Массивы	123
10.1. Создание массива	123
10.2. Ассоциативные и индексные массивы	127
10.3. Многомерные массивы	128
10.4. Интерполяция элементов массива в строки	130
10.5. Конструкция <i>list()</i>	131
10.6. Обход массива	132
10.7. Цикл <i>foreach</i>	132
10.8. Слияние массивов	134
10.9. Сравнение массивов	136
10.10. Проверка существования элементов массива	138
10.11. Удаление элементов массива	141
Задания	142
Глава 11. Функции	143
11.1. Объявление и вызов функции	143
11.2. Параметры и аргументы функции	146
11.3. Типы параметров и возвращаемого значения	147
11.4. Передача параметров по значению и ссылке	147
11.5. Необязательные параметры	148
11.6. Переменное количество параметров	149
11.7. Глобальные переменные	150
11.8. Статические переменные	151
11.9. Возврат массива функцией	152
11.10. Рекурсивные функции	152
11.11. Вложенные функции	154
11.12. Динамическое имя функции	154
11.13. Анонимные функции	155
11.14. Замыкания	157
Задания	158
Глава 12. Строковые функции	159
12.1. Строки как массивы	159
12.2. UTF-8. Расширение <i>mbstring</i>	160

12.3. Функции для работы с символами	162
12.4. Поиск в строке	163
12.5. Замена в тексте.....	164
12.6. Работа с HTML-кодом.....	166
12.7. Форматный вывод.....	170
12.8. Объединение и разбиение строк.....	173
12.9. Сериализация объектов и массивов	175
12.10. JSON-формат.....	175
Задания	179

Глава 13. Взаимодействие PHP с HTML 181

13.1. Передача параметров методом GET	181
13.2. HTML-форма и ее обработчик	184
13.3. Текстовое поле.....	188
13.4. Поле для приема пароля.....	189
13.5. Текстовая область	190
13.6. Скрытое поле	191
13.7. Флажок	193
13.8. Список	195
13.9. Переключатель.....	197
13.10. Загрузка файла на сервер	198
13.11. Переадресация	201
Задания	204

Глава 14. Суперглобальные массивы 205

14.1. Типы суперглобальных массивов.....	205
14.2. Cookie	206
14.3. Сессии.....	208
14.4. Переменные окружения	210
14.5. Массив \$_SERVER	212
14.5.1. Элемент \$_SERVER['DOCUMENT_ROOT']	212
14.5.2. Элемент \$_SERVER['HTTP_ACCEPT']	212
14.5.3. Элемент \$_SERVER['HTTP_ACCEPT_LANGUAGE']	213
14.5.4. Элемент \$_SERVER['HTTP_HOST']	214
14.5.5. Элемент \$_SERVER['HTTP_REFERER']	214
14.5.6. Элемент \$_SERVER['HTTP_USER_AGENT']	214
14.5.7. Элемент \$_SERVER['REMOTE_ADDR']	214
14.5.8. Элемент \$_SERVER['SCRIPT_FILENAME']	215
14.5.9. Элемент \$_SERVER['SERVER_NAME']	215
14.5.10. Элемент \$_SERVER['REQUEST_METHOD']	216
14.5.11. Элемент \$_SERVER['QUERY_STRING']	216
14.5.12. Элемент \$_SERVER['PHP_SELF']	217
14.5.13. Элемент \$_SERVER['REQUEST_URI']	217
Задания	217

Глава 15. Фильтрация и проверка данных..... 219

15.1. Фильтрация или проверка?	219
15.2. Фильтры проверки.....	221

15.3. Фильтры очистки	224
15.4. Пользовательская фильтрация данных	227
15.5. Фильтрация внешних данных	228
Задания	230
Глава 16. Методы	231
16.1. Определение метода	231
16.2. Обращение к переменным объекта	232
16.3. Статические методы	234
16.4. Ключевое слово <i>self</i>	234
Задания	235
Глава 17. Специальные методы	237
17.1. Конструктор <i>__construct()</i>	237
17.2. Параметры конструктора	239
17.3. Деструктор <i>__destruct()</i>	241
17.4. Методы-аксессоры <i>__set()</i> и <i>__get()</i>	242
17.5. Динамические методы	244
17.6. Интерполяция объекта	246
Задания	248
Глава 18. Наследование	249
18.1. Наследование	249
18.2. Спецификаторы доступа и наследование	250
18.3. Перегрузка методов	253
18.4. Позднее статическое связывание	255
18.5. Полиморфизм	257
18.6. Абстрактные классы	259
18.7. Абстрактные методы	260
18.8. <i>Final</i> -методы класса	261
18.9. <i>Final</i> -классы	262
18.10. Анонимные классы	262
18.11. Оператор <i>instanceof</i>	264
Задания	264
Глава 19. Интерфейсы	265
19.1. Ограничения наследования	265
19.2. Создание интерфейса	269
19.3. Наследование интерфейсов	271
19.4. Реализация нескольких интерфейсов	274
19.5. Реализует ли объект интерфейс?	276
Задание	277
Глава 20. Трейты	279
20.1. Создание трейта	279
20.2. Трейты и наследование	282
20.3. Разрешение конфликтов	285
20.4. Вложенные трейты	287
Задание	288

Глава 21. Исключения	289
21.1. Синтаксис исключений	290
21.2. Интерфейс класса <i>Exception</i>	291
21.3. Генерация исключений в классах	293
21.4. Создание собственных исключений.....	296
21.5. Перехват исключений производных классов	299
21.6. Повторная генерация исключений	300
21.7. Блок <i>finally</i>	302
Задание	303
Глава 22. Ошибки	305
22.1. Ошибки и исключения	305
22.2. Типы уведомлений.....	307
22.3. Пользовательские ошибки	309
22.4. Подавление ошибок.....	310
Задания	311
Глава 23. Пространство имен	313
23.1. Создание пространства имен	313
23.2. Иерархия пространств имен	318
23.3. Глобальное пространство имен	319
23.4. Текущее пространство имен	319
23.5. Импортирование	320
Задания	321
Глава 24. Автозагрузка	323
24.1. Функция <i>__autoload()</i>	323
24.2. Функция <i>spl_autoload_register()</i>	326
Задание	327
Глава 25. Шаблоны проектирования	329
25.1. Зачем нужны шаблоны проектирования?	330
25.2. Одиночка (Singleton)	331
25.3. Фабричный метод (Factory Method)	332
25.4. Модель-Представление-Контроллер.....	338
Задания	349
Глава 26. Компоненты	351
26.1. Composer: управление компонентами.....	351
26.2. Установка Composer	352
26.2.1. Установка в Windows	352
26.2.2. Установка в Mac OS X.....	354
26.2.3. Установка в Ubuntu.....	354
26.3. Где искать компоненты?	354
26.4. Установка компонента	355
26.5. Использование компонента	357
Задания	358

Глава 27. База данных PostgreSQL	359
27.1. Почему PostgreSQL?	360
27.2. Установка PostgreSQL	361
27.2.1. Установка в Windows	361
27.2.2. Установка в Mac OS X	363
27.2.3. Установка в Ubuntu	363
27.3. Введение в СУБД и SQL	364
27.4. Первичные ключи	366
27.5. Управление базами данных	368
27.6. Управление таблицами	370
27.7. Вставка записей в таблицу	371
27.8. Удаление записей	372
27.9. Обновление записей	373
27.10. Выборка записей	374
Задания	375
Глава 28. PHP-расширение PDO	377
28.1. Настройка расширения PDO	377
28.2. Установка соединения с базой данных	378
28.3. Выполнение SQL-запросов	379
28.4. Обработка ошибок	380
28.5. Извлечение данных	382
28.6. Параметризация SQL-запросов	384
Задания	385
Глава 29. NoSQL база данных Redis	387
29.1. Почему Redis?	388
29.2. Установка сервера	389
29.2.1. Установка в среде Ubuntu	389
29.2.2. Установка в среде Mac OS X	389
29.2.3. Установка в Windows	390
29.2.4. Проверка работоспособности	390
29.3. Клиент <i>redis-cli</i>	391
29.4. Вставка и получение значений	392
29.5. Обновление и удаление значений	393
29.6. Управление ключами	395
29.7. Время жизни ключа	395
29.8. Типы данных	396
29.9. Хэш	397
29.10. Множество	398
29.11. Отсортированное множество	400
29.12. Базы данных	402
29.13. Производительность Redis	402
Задания	403
Глава 30. PHP-расширение Redis	405
30.1. Установка расширения <i>php-redis</i>	405
30.2. Хранение сессий в Redis	406

30.3. Методы для обслуживания данных в Redis	407
30.4. Кэширование данных	409
Задания	414
Глава 31. Итераторы.....	415
31.1. Интерфейсы для создания итераторов	415
31.2. Интерфейс <i>ArrayAccess</i>	419
31.3. Класс <i>ArrayObject</i>	422
31.4. Класс <i>DirectoryIterator</i>	423
31.5. Класс <i>FilterIterator</i>	424
31.6. Класс <i>LimitIterator</i>	425
31.7. Рекурсивные итераторы	426
Задания	426
Глава 32. Генераторы и итераторы	427
32.1. Отложенные вычисления	427
32.2. Манипуляция массивами.....	430
32.3. Экономия ресурсов.....	432
32.4. Использование ключей.....	433
32.5. Связь генераторов с объектами	435
Задания	436
Заключение.....	437
Предметный указатель	439

Предисловие

Вы держите в руках четвертое, полностью переработанное издание популярной книги. Предыдущее издание было посвящено PHP 5 и вышло 8 лет назад. С тех пор язык обогатился большим количеством нововведений, все их мы осветим на страницах книги.

В связи с этим книгу пришлось полностью переписать, лишь 10 глав из 32-х основаны на материале предыдущей книги, хотя и их пришлось подвергнуть глубокой переработке. Остальные главы написаны с чистого листа.

Объектно-ориентированный подход

До версии PHP 5 поддержка объектно-ориентированного программирования (ООП) в языке была довольно скудна. В PHP 5 объектно-ориентированный подход долго оставался альтернативой для традиционного процедурного подхода. PHP 7 практически полностью предназначен для объектно-ориентированной разработки.

Без ООП невозможна разработка современных PHP-приложений: все больше расширений предполагают объектно-ориентированный интерфейс, компоненты оформляются в виде классов, PSR-стандарты и современные фреймворки диктуют разработку, полностью ориентированную на объектно-ориентированный подход.

Именно поэтому знакомиться с объектно-ориентированным подходом мы начинаем сразу с первых глав книги. Рассматриваем шаблоны проектирования и там, где это возможно, ориентируемся на новые расширения с объектно-ориентированным интерфейсом.

PostgreSQL и Redis

Традиционно PHP-приложения работают совместно с СУБД MySQL. В редкой книге по PHP не уделяется внимание этой популярной базе данных. Однако мы отступим от традиции.

С одной стороны, перепродажа компании AB MySQL корпорации Sun, которая в свою очередь была поглощена в 2009 г. Oracle, привела к тому, что наиболее по-

пулярная свободная СУБД оказалась в руках крупнейшего в мире производителя коммерческих баз данных. Развитие MySQL если и не находится в стагнации, то значительно уступает конкурирующим СУБД, например PostgreSQL, которая долгое время оставалась на вторых ролях.

С другой стороны, развитие объектно-ориентированного подхода и увеличение количества памяти на серверах привели к всплеску интереса к нереляционным базам данных, которые оформились в виде движения NoSQL. PostgreSQL — не совсем традиционная СУБД. Задуманная как объектно-ориентированная база данных, PostgreSQL за несколько лет превзошла NoSQL-движение. Поэтому уже сегодня можно наблюдать, как большинство современных Web-приложений переходят на PostgreSQL.

Кроме этого, вместо традиционного memcached мы рассмотрим NoSQL базу данных Redis. Как правило, современные Web-приложения не обходятся без одной или нескольких NoSQL баз данных, поэтому обойти их вниманием не представляется возможным. Redis выделяется среди них высокой производительностью (100 000 RPS — request per seconds), богатыми возможностями (коллекции, кластеризация, механизм Pub/Sub).

Задания

Мы не рассматриваем детальную установку полноценного окружения, включающего Web-сервер, настройку его связи с PHP, обеспечение безопасного доступа к удаленному серверу. Вместо этого используется встроенный PHP-сервер, сразу же доступный при установке PHP.

В силу ограничений по объему книги пришлось отказаться от описания многих особенностей PHP, например: детального рассмотрения сокетов, расширения curl, математических функций, преобразования изображений, управления буфером вывода, регулярных выражений и даже файловых функций. Все эти возможности выходят за рамки книги.

Вместо этого каждая глава снабжается заданиями, которые побуждают исследовать документацию, знакомиться с не вошедшими в книгу функциями и расширениями, исследовать альтернативные базы данных, читать статьи, искать решение. Всего 100 заданий.

Исходные коды

В начале каждой главы (за исключением 1-й и 29-й) указан каталог, в котором можно обнаружить примеры данной главы, названия файлов приведены в заголовках к листингам.

Исходные коды к книге можно найти на GitHub-аккаунте по адресу:

<https://github.com/igorsimdyanov/phpworkshop>

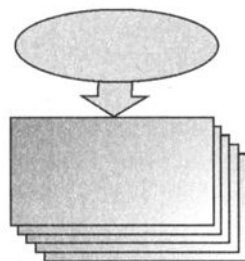
По ссылке <https://github.com/igorsimdyanov/phpworkshop/issues> можно адресовать вопросы авторам.

Электронный архив с исходными кодами к книге также можно скачать по ссылке <ftp://ftp.bhv.ru/9785977538176.zip>. Эта ссылка доступна и со страницы книги на сайте www.bhv.ru.

Благодарности

С Максимом Кузнецовым мы написали множество книг, и первое издание этой книги было стартом нашей совместной работы. Пять лет, как его с нами нет: жил быстро, ярко, помогал многим, казалось бы, в безвыходных ситуациях. Это издание книги посвящено ему.

ГЛАВА 1



Что представляет собой PHP?

Язык программирования PHP — серверный язык, при помощи которого можно создавать Web-сайты, причем как небольшие лендинги, состоящие из одной страницы, так и гигантские системы, использующие сотни и тысячи серверов. Электронная энциклопедия Wikipedia, социальные сети Facebook, "ВКонтакте", электронная площадка объявлений Avito созданы с использованием PHP.

Будучи одним из самых первых языков программирования, ориентированных на Web-разработку, PHP прошел длительный путь практически с самого начала зарождения Web. Поэтому в мире он остается одним из самых популярных и востребованных языков.

1.1. Достоинства и недостатки

В основе популярности PHP лежат следующие достоинства.

- ☐ Ориентация на Web-разработку — PHP создавался, развивался и поддерживается как язык для создания Web-сайтов. Многие конструкции и решения в нем созданы для удобства работы в Web-среде.
- ☐ Кроссплатформенность — PHP перенесен на все основные операционные системы: можно разрабатывать сайт в Windows, Mac OS X, а эксплуатировать на Linux-сервере. Сложности переноса будут минимальны и нивелироваться языком.
- ☐ Бесплатность — PHP является разработкой из мира свободного программного обеспечения, не потребуется платить ни за сам язык, ни за большинство сопутствующих программ (редакторы, Web-серверы, базы данных). Вдобавок большинство программных продуктов, с которыми придется иметь дело, будут иметь доступный для изучения и модификации исходный код. Вложения могут потребоваться при аренде доменного имени и сервера для публикации сайта в Интернете. Однако изучать PHP можно, не вкладывая ни копейки.
- ☐ Низкий порог входа — изучить PHP и начать создавать на нем готовые приложения много проще, чем с использованием конкурирующих технологий (.NET,

Python, Ruby, Go). Изучение PHP не закрывает для разработчика другие технологии, в Web сам язык — значительная, но меньшая часть используемых технологий. Знания, приемы работы, сопутствующие технологии (Web-серверы, базы данных, библиотеки, вспомогательные языки) пригодятся и в любой другой экосистеме, отличной от PHP. При создании собственного бизнеса собрать команду PHP-разработчиков зачастую проще и дешевле всего.

По закону сохранения, любая вещь, обладающая хоть каким-либо достоинством, имеет недостатки. Ими обладает и PHP.

- ❑ Отсутствие лидера — многие технологии и языки имеют лидера, архитектора, который определяет облик технологии, задает вектор развития, принимает решение о том, что должно быть обязательно, а чего не будет никогда (Linux, Python, Ruby и т. п.). В PHP лидера нет, многие решения и конструкции — это компромисс заинтересованных групп и исторически сложившихся реалий.
- ❑ Непоследовательный синтаксис — при изучении языка PHP, особенно старой части, основанной на функциях, можно заметить, что часть функций имеет префиксы `array_`, `str_`, часть не имеет. Параметры функций могут быть расположены не совсем логично и не так, как в другой функции этой же группы.
- ❑ PHP — уже довольно долго живущий язык. Когда язык только появляется, он довольно элегантный и внутренне согласованный. По мере жизненного цикла язык обрастает дополнительными ключевыми словами, артефактами, устаревшими конструкциями, которые вроде есть, работают, но которыми не рекомендуется пользоваться. У PHP была довольно бурная молодость, в ходе которой была отменена масса директив и приемов, которые на первый взгляд должны были облегчать разработку, а на практике оборачивались серьезными проблемами безопасности. Сам PHP, стартовавший как неobjектно-ориентированный язык, в настоящий момент стал полноценным объектно-ориентированным языком. Однако в нем полно старых процедурных артефактов, которыми придется пользоваться.
- ❑ Сообщество PHP-разработчиков велико и разъединено, т. к. PHP — это одна из первых технологий для разработки Web-проектов, половина Интернета создана с его участием. В PHP-разработку одновременно было вовлечено огромное количество программистов по всему миру. Все это породило большое число самых разных подходов, фреймворков и не совместимых друг с другом экосистем. Более того, благодаря усилиям мощных и влиятельных социальных сетей (в первую очередь Facebook, "ВКонтакте") появились альтернативные реализации PHP. Это плохо, т. к. многие экосистемы внутри PHP не совместимы, а сообщество раздроблено и тратит силы на создание одних и тех же библиотек в рамках разных групп. Ситуация исправляется и при помощи PSR-стандартов. Разработчики договариваются о единых правилах и интерфейсах, обеспечивающих совместимость фреймворков, но этот процесс еще в начале пути, в то время как конкурирующие технологии (.NET, Ruby) уже имеют единую платформу для всех фреймворков.

1.2. Структура PHP

Язык PHP имеет ядро и расширения языка. Между ядром и расширениями довольно трудно провести границу, т. к. многие расширения давно включены в состав ядра или распространяются в виде скомпилированных бинарных библиотек и легко устанавливаются.

Есть и другая часть — код, созданный на PHP, который условно можно поделить на следующие типы:

- ❑ компоненты — библиотеки на PHP, которые собираются при помощи менеджера пакетов Composer (см. главу 26);
- ❑ фреймворки — готовые сборки, зачастую из компонентов, при помощи которых можно создавать сайты любой степени сложности. В книге, к сожалению, мы их не касаемся, однако если вы выберете PHP в качестве основного языка разработки, то не пройдете мимо них. На следующие PHP-фреймворки стоит обратить внимание: Symfony, Laravel, Zend, Yii. Всего их сотни, если не тысячи;
- ❑ готовые приложения — готовые к использованию разработки на PHP. Это системы управления контентом (Wordpress, Drupal), форумы (phpBB), Web-интерфейсы управления базами данных (phpMyAdmin, pgAdmin).

Книга, которую вы держите в руках, познакомит вас с языком; описанные выше системы созданы с использованием языка PHP, но требуют отдельного изучения, и, к сожалению, их обсуждение выходит за рамки нашей книги.

1.3. Сопутствующие технологии

При помощи PHP можно быстро разрабатывать Web-сайты, однако современные реализации PHP — это не самая быстрая и эффективная часть сайта. Поэтому для запуска сайта потребуются дополнительное программное обеспечение и технологии.

- ❑ Web-сервер — программа, которая обеспечивает взаимодействие клиента и вашего приложения посредством протокола HTTP. На протяжении всей книги мы используем встроенный PHP-сервер (см. главу 3), хотя для эксплуатации настоящего сайта потребуется Web-сервер nginx или Apache.
- ❑ Сервер базы данных — данные нужно где-то хранить. В книге довольно подробно рассматриваются две базы данных: PostgreSQL (см. главы 27–28) и Redis (см. главы 29–30). Однако это далеко не все базы данных, которые вам встретятся на практике, да и каждая из затронутых заслуживает отдельной книги.
- ❑ Система контроля версий Git, которая служит для хранения истории разработки, резервного копирования, доставки кода на сервер, организации командной работы. Работаете ли вы в коллективе или в одиночку — Git в настоящий момент превратился в основной инструмент современного программиста, какой бы язык программирования не был выбран в качестве базового.

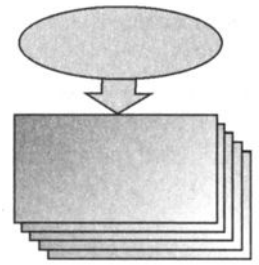
Этот список можно продолжать и продолжать. Однако на самом деле можно начать даже без этого — опираясь просто на язык PHP. Если вам интересно, что вас ждет после изучения PHP, загляните в заключение.

Задание

Установите систему Git и загрузите исходные коды к книге с GitHub:

<https://github.com/igorsimdyanov/phpworkshop>.

ГЛАВА 2



Установка PHP

Листинги данной главы можно найти
в подкаталоге `buildin`.

В текущей главе будет рассмотрен порядок установки PHP в операционных системах Windows, Mac OS X и Linux (дистрибутив Ubuntu). Наличие PHP-интерпретатора позволит запускать PHP-программы, однако для запуска Web-приложений потребуется также Web-сервер. В ранних версиях PHP для этих целей использовался промышленный Web-сервер: Apache или nginx. Начиная с версии 5.4, дистрибутив PHP снабжается встроенным сервером, возможностей которого вполне хватает для локальной разработки и тестирования Web-приложений.

2.1. Установка в Windows

Для загрузки Windows-дистрибутива следует посетить раздел загрузки бинарных файлов официального сайта PHP: <http://windows.php.net/download>. Каждый релиз снабжается четырьмя вариантами:

- ☐ **x86 Non Thread Safe** — 32-битный CGI-вариант дистрибутива;
- ☐ **x86 Thread Safe** — 32-битный вариант для установки в качестве модуля Web-сервера;
- ☐ **x64 Non Thread Safe** — 64-битный CGI-вариант дистрибутива;
- ☐ **x64 Thread Safe** — 64-битный вариант для установки в качестве модуля Web-сервера.

Вариант **Thread Safe** предназначен для безопасного выполнения PHP в параллельных потоках в рамках одного системного процесса, например, если PHP устанавливается в качестве модуля Web-сервера Apache. Так как мы собираемся использовать встроенный сервер, не имеет значения, какой дистрибутив будет выбран, лучше всего воспользоваться вариантом **Non Thread Safe**. Последний вариант так же применяется при подключении PHP в качестве внешнего FastCGI-приложения, которое запускается на каждый внешний запрос.

Перед названием дистрибутива может быть помещена одна из аббревиатур VC11, VC14, означающих версии Visual Studio (2012 и 2015, соответственно), при помощи которой был скомпилирован дистрибутив. Для того чтобы успешно запустить проект, следует загрузить соответствующий распространяемый пакет Visual C++ для Visual Studio, который содержит необходимые динамические библиотеки:

❑ вариант VC11

<http://www.microsoft.com/en-us/download/details.aspx?id=30679>;

❑ вариант VC14

<http://www.microsoft.com/en-us/download/details.aspx?id=48145>.

ВНИМАНИЕ!

Необходимы библиотеки именно от английского варианта Visual Studio, русский вариант пакета не подойдет.

После загрузки zip-архива его следует распаковать в какую-нибудь папку, например C:\php.

Убедиться в том, что PHP доступен, можно, запустив командную строку, а затем перейти в папку C:\php при помощи команды¹

```
> cd C:\php
```

Выполнив в командной строке команду `php` с параметром `-v`, можно узнать текущую версию PHP:

```
> php -v
```

```
PHP 7.0.0 (cli) (built: Dec 3 2015 09:31:54) ( NTS )
Copyright (c) 1997-2015 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2015 Zend Technologies
```

Для того чтобы команда PHP была доступна в любой точке файловой системы, путь к PHP-интерпретатору следует прописать в переменной окружения `PATH`.

Для доступа к переменным окружения нужно открыть Панель управления, перейти к разделу **Система**. Самый быстрый способ добраться до этого пункта — это щелкнуть правой кнопкой мыши по кнопке **Пуск** и выбрать пункт **Система** из контекстного меню. В операционных системах, предшествующих Windows 8, следует выбрать в меню **Пуск** пункт **Компьютер** и в контекстном меню выбрать пункт **Свойства**. В открывшемся окне Панели управления с активным разделом **Система** слева щелкнуть по ссылке **Дополнительные параметры системы**. Затем в окне **Свойства системы** на вкладке **Дополнительно** необходимо нажать кнопку **Переменные среды**. В открывшемся диалоговом окне в разделе **Системные переменные** следует отыскать переменную окружения `PATH` и дополнить ее путем к каталогу C:\php. Отдельные пути в значении переменной `PATH` разделяются точкой с запятой (в конце всей строки точка с запятой не требуется). После этого команда `php` будет доступна в любой папке компьютера.

¹ Полужирным шрифтом будем выделять команды или текст, вводимый пользователем.

2.2. Установка в Mac OS X

Прежде чем устанавливать PHP, следует установить Command Line Tools for XCode из магазина AppStore. XCode — это интегрированная среда разработки приложений для Mac OS X и iOS. Полная загрузка XCode не обязательна, достаточно установить инструменты командной строки и компилятор. Убедиться в том, установлен ли XCode, можно при помощи команды

```
$ xcode-select -p
/Applications/Xcode.app/Contents/Developer
```

Если вместо указанного выше пути выводится предложение установить Command Line Tools, следует установить этот пакет, выполнив команду

```
$ xcode-select --install
```

Теперь можно приступить к установке PHP, для чего лучше всего воспользоваться менеджером пакетов Homebrew. На момент написания книги установить Homebrew можно было при помощи команды

```
$ ruby -e "$(curl -fsSL
↳ https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Впрочем, точную команду всегда можно выяснить на официальном сайте <http://brew.sh>. После установки в командной строке будет доступна команда `brew`, при помощи которой можно загружать, удалять и обновлять пакеты с программным обеспечением.

Сразу после установки будет не лишним установить дополнительные библиотеки, которые могут потребоваться расширениям PHP:

```
$ brew install freetype jpeg libpng gd zlib
```

Для того чтобы получить доступ к репозиториям с PHP-дистрибутивами, следует выполнить серию команд:

```
$ brew tap homebrew/dupes
$ brew tap homebrew/versions
$ brew tap homebrew/homebrew-php
```

По умолчанию команда `brew tap` предполагает, что ей передаются названия GitHub-репозитория. Таким образом, предыдущие команды прописывают доступ к следующим репозиториям:

<https://github.com/Homebrew/homebrew-dupes>

<https://github.com/Homebrew/homebrew-versions>

<https://github.com/Homebrew/homebrew-php>

Можно самостоятельно найти на GitHub дистрибутив или альтернативный проект и добавить его в менеджер пакетов Homebrew. Если пакет больше не нужен, его можно исключить при помощи команды `brew untap`. Разумеется, по возможности лучше использовать официальные пакеты Homebrew.

После выполнения приведенных выше команд надо убедиться в том, что пакеты успешно добавлены. Для этого следует выполнить команду `brew tap` без параметров. В результате будет выведен список добавленных репозиториев.

```
$ brew tap
homebrew/dupes/homebrew/php
homebrew/versions
```

Теперь можно установить PHP, запустив команду установки:

```
$ brew install php71
```

После установки PHP готов к работе.

```
$ php -v
PHP 7.1.2 (cli) (built: Feb 17 2017 10:51:21) ( NTS )
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.1.0, Copyright (c) 1998-2017 Zend Technologies
```

Если PHP недоступен, следует убедиться, что путь к `sbin`-каталогу Homebrew прописан в переменной окружения `PATH` в файле `~/.bash_profile`:

```
# Homebrew sbin path
PATH=/usr/local/sbin:$PATH
```

2.3. Установка в Linux (Ubuntu)

Для того чтобы установить PHP в Ubuntu, следует воспользоваться штатным менеджером пакетов `apt-get`. Предварительно нужно обновить сведения о репозиториях и текущие пакеты при помощи команд:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Затем можно приступить к установке:

```
$ sudo apt-get install php7.0
```

После успешной установки команда `php` должна быть доступна в любой точке компьютера:

```
$ php -v
PHP 7.0.15-0ubuntu0.16.04.4 (cli) ( NTS )
Copyright (c) 1997-2017 The PHP Group
Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies
    with Zend OPcache v7.0.15-0ubuntu0.16.04.4, Copyright (c) 1999-2017,
    by Zend Technologies
```

2.4. Встроенный сервер

Для проверки работоспособности интерпретатора PHP создадим скрипт `index.php`, который выводит традиционную для книг по программированию фразу "Hello, world!" (листинг 2.1).

Листинг 2.1. Проверочный скрипт Hello World. Файл index.php

```
<?php
echo 'Hello, world!';
?>
```

Если у текущего пользователя отсутствуют права администратора, тогда в папке со скриптом `index.php` следует выполнить команду

```
php -S localhost:4000
```

Команда запустит на 4000-м порту Web-сервер. Обратившись в браузере по адресу **http://localhost:4000/**, можно увидеть фразу "Hello, world!". Если работа ведется из-под учетной записи системного администратора (Windows) или задействована команда `sudo` (Mac OS X или Linux), встроенный сервер можно запустить на стандартном 80-м порту:

```
php -S localhost:80
```

В этом случае порт в адресе можно не указывать, браузер автоматически будет обращаться по 80-порту, закрепленному за протоколом HTTP: **http://localhost/**. По умолчанию в качестве корневого каталога выступает текущая папка, именно в ней будет произведен поиск индексного файла `index.php`. Однако при помощи параметра `-t` можно указать произвольную папку:

```
php -S localhost:4000 -t buildin
```

Журнальные записи или, как еще говорят, логи сервера выводятся непосредственно в консоль, в которой он был запущен. Остановить сервер можно, нажав комбинацию клавиш `<Ctrl>+<C>`.

2.5. Файл hosts

В предыдущем разделе в качестве домена использовался `localhost`, который является псевдонимом для IP-адреса 127.0.0.1. На локальном хосте соответствие псевдонима IP-адресу прописывается в файле `hosts`, который в UNIX-подобной операционной системе можно обнаружить по пути `/etc/hosts`, а в Windows по пути `C:\Windows\system32\drivers\etc\hosts`.

Как правило, в файле присутствуют как минимум две записи, сопоставляющие домену `localhost` локальный IP-адрес 127.0.0.1 в IPv4- и IPv6-форматах:

```
127.0.0.1      localhost
::1           localhost
```

Именно наличие этих записей позволило нам запустить встроенный сервер с использованием в качестве домена `localhost`. Для того чтобы настроить альтернативные псевдонимы, можно добавить дополнительные записи:

```
127.0.0.1      site.dev
127.0.0.1      www.site.dev
```



```
127.0.0.2    project.dev
127.0.0.2    www.project.dev
```

IP-адреса, начинающиеся со 127, предназначены для локального использования, поэтому для тестирования собственных проектов вы можете назначать любые адреса из этого диапазона.

После добавления новых псевдонимов в файл `hosts` их можно использовать совместно со встроенным сервером, например **`http://site.dev:4000/`**.

2.6. Вещание вовне

При использовании локальных IP-адресов из диапазона `127.XXX` можно быть уверенным, что никто извне не сможет обратиться к серверу. Если же наоборот требуется продемонстрировать результат работы вашего приложения, в качестве хоста можно указать IP-адрес `0.0.0.0`:

```
php -S 0.0.0.0:4000
```

В этом случае можно получить доступ к Web-серверу, обратившись по IP-адресу хоста, где запущен сервер, например **`http://192.168.0.1:4000`**. Для того чтобы можно было обратиться к хосту по псевдониму, либо его придется прописать в `hosts`-файле каждого компьютера, с которого идет обращение к серверу, либо зарегистрировать доменное имя и связать его с IP-адресом компьютера, на котором сервер запущен. Разумеется, в этом случае при запуске в качестве хоста потребуется указать это доменное имя.

```
php -S example.com:80
```

Впрочем, встроенный сервер предназначен лишь для разработки, для обеспечения работы полноценного сайта лучше воспользоваться промышленными серверами, такими как Apache или nginx.

2.7. Настройка PHP

PHP имеет огромное количество разнообразных настроек, которые сосредоточены в файле `php.ini`. Сразу после установки вместо файла `php.ini` можно обнаружить лишь два файла:

- ☐ `php.ini-production` — рекомендованный набор параметров для рабочего сервера;
- ☐ `php.ini-development` — рекомендованный набор параметров для рабочей станции разработчика.

Для локальной разработки файл `php.ini-development` следует переименовать в `php.ini`. По умолчанию интерпретатор PHP последовательно ищет конфигурационный файл в следующих местах:

- ☐ по пути, указанному в переменной окружения `PHPRC`;
- ☐ в текущем каталоге (если скрипт выполняется под управлением Web-сервера);

❑ в каталоге C:\Windows\ в случае Windows, в каталоге /etc/ или /etc/php7/ в Linux, в каталоге /usr/local/etc/php7.0 в Mac OS X или в каталоге компиляции в случае любой другой операционной системы.

Если имеется несколько файлов `php.ini` и нет уверенности в том, какой из них используется в настоящий момент, выяснить путь к используемому файлу `php.ini` можно из отчета информационной функции `phpinfo()` (листинг 2.2).

Листинг 2.2. Информация о PHP. Файл `phpinfo.php`

```
<?php
phpinfo();
?>
```

Функция выводит подробную информацию обо всех параметрах PHP. Путь к файлу `php.ini`, который сейчас используется PHP, указывается в параметре `Loaded Configuration File`.

Можно явно указать PHP местоположение файла `php.ini` при помощи параметра `-c`. Для Windows команда может выглядеть следующим образом:

```
php -s 127.0.0.1:4000 -c C:\php\php.ini
```

Для UNIX-подобной операционной системы:

```
php -s 127.0.0.1:4000 -c /etc/php.ini
```

Конфигурационный файл PHP является обычным текстовым файлом, который можно открыть при помощи любого редактора. Содержимое файла `php.ini` состоит из секций и директив. Секции заключаются в квадратные скобки, например `[PHP]`, после которых следуют директивы, имеющие такой формат:

```
directive = value
```

Здесь *directive* — это название директивы, а *value* — ее значение. Все строки, в начале которых располагается точка с запятой (;), считаются комментариями и игнорируются.

Допускается не указывать значение *value*. В этом случае директива инициализируется пустой строкой. Этого же результата можно добиться, присвоив ей значение `none`.

Сразу после установки PHP следует отредактировать как минимум одну директиву — текущий часовой пояс. Для того чтобы выставить московский часовой пояс, необходимо найти директиву `date.timezone` и привести ее к виду:

```
date.timezone = 'Europe/Moscow'
```

Если встроенный сервер при этом оставался запущенным, потребуется его перезапуск для того, чтобы изменения в `php.ini` вступили в силу.

2.8. Расширения

Интерпретатор PHP строится по модульному принципу. Язык состоит из ядра, к которому при необходимости могут подключаться расширения — библиотеки, дополняющие язык новыми возможностями. Например, для обработки изображений можно подключить расширение GDLib, для связи с базами данных — расширение PDO, для поддержки многобайтовых строк (UTF-8) — mbstring. Часть расширений получили настолько широкую популярность, что их включили в состав ядра. Например, это строковые функции из расширения calendar или поддержка сессий из расширения session.

ВНИМАНИЕ!

Материал текущего раздела можно пропустить при первом чтении.

Каждое расширение увеличивает размер оперативной памяти, которую занимает интерпретатор PHP. Чем больше памяти потребляет интерпретатор, тем меньше процессов PHP можно запустить на сервере, тем меньше соединений сервер может обслужить в одну секунду. Поэтому расширения подключаются и отключаются по мере необходимости. Для того чтобы уменьшить размер исполняемого файла PHP, а следовательно, и объем потребляемой оперативной памяти, большинство расширений не включаются в состав ядра PHP. Вместо этого они компилируются в виде внешних динамических библиотек: *.dll для Windows и *.so для UNIX-подобных операционных систем.

Самый простой способ выяснить, подключено ли расширение, — это выполнить команду `php -m`, которая выводит список доступных расширений. Можно так же воспользоваться отчетом функции `phpinfo()` (см. листинг 2.2). В разделе Configuration указываются подключенные расширения, в таблицах, которые сопровождают каждое из расширений, приводится список параметров данного расширения. Большинство из этих параметров могут быть скорректированы в конфигурационном файле `php.ini`.

В дистрибутивах для операционной системы Windows расширения, оформленные в виде динамических библиотек, скомпилированы, но не подключены. Обнаружить их можно в подкаталоге `ext` основной папки PHP-дистрибутива:

```
php_bz2.dll
php_com_dotnet.dll
php_curl.dll
...
php_tidy.dll
php_xmlrpc.dll
php_xsl.dll
```

Для того чтобы подключить одно из таких внешних расширений, необходимо отредактировать конфигурационный файл `php.ini`. Путь к нему можно обнаружить в отчете функции `phpinfo()`. В конфигурационном файле `php.ini` следует найти директиву `extension_dir` и указать в ней путь к папке с расширениями:

```
extension_dir = "ext"
```

После того как путь к папке с расширением указан, можно активировать сами расширения, воспользовавшись директивой `extension`. В конфигурационном файле `php.ini`, как правило, уже добавлены закомментированные директивы для всех расширений из папки `ext`. Нужно расширение необходимо активировать, убрав точку с запятой из начала строки:

```
extension=php_mbstring.dll
```

В случае UNIX-подобных операционных систем установка расширений осуществляется еще проще. Как правило, они оформлены в виде отдельных пакетов.

Так в Mac OS X, в основном, используется менеджер пакетов Homebrew. В *разд. 2.2* PHP устанавливался при помощи команды

```
$ brew install php71
```

Точно так же устанавливается расширение, только название пакета `php71` дополняется именем расширения, например, для CURL команда может выглядеть следующим образом:

```
$ brew install php71-curl
```

В современных Linux-дистрибутивах установка расширений также осуществляется при помощи менеджера пакетов, в случае Ubuntu это `apt-get`. Напомним, что установка PHP осуществляется командой

```
$ sudo apt-get install php7.0
```

Точно так же устанавливаются расширения. Так, CURL можно установить при помощи команды

```
$ sudo apt-get install php7.0-curl
```

2.9. Документация

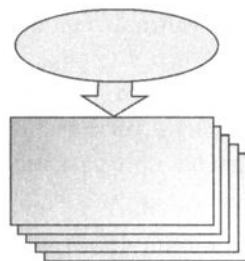
Главным источником информации о PHP для всех разработчиков является официальный сайт <http://php.net>. На его страницах можно обнаружить раздел документации, в том числе на русском языке. К сожалению, перевод на русский язык иногда не полон и отстает от английской версии. Мы не сможем осветить на страницах книги все возможности языка, нам просто не хватит для этого места, поэтому вам придется обращаться к документации PHP для того, чтобы воспользоваться всеми возможностями PHP.

Задания

1. При помощи `php -h` выведите справочную информацию о параметрах интерпретатора PHP. Найдите способ запуска интерактивного режима и выполните в нем фразу "Hello world".

2. Найдите в сети редактор Sublime Text и установите его в своей операционной системе. Создайте в нем проект Hello World с программой из листинга 2.1.
3. На сайте **<http://php.net>** воспользуйтесь поиском, найдите информацию о конструкции `echo` и функции `phpinfo()`.
4. Установите VirtualBox и запустите в нем альтернативную операционную систему: Ubuntu, если вы работаете в Windows или Mac OS X, или Windows, если ваша рабочая система — Ubuntu.
5. Установите Web-сервер nginx и запустите php-fpm, настройте их на совместную работу таким образом, чтобы они поддерживали обработку PHP-скриптов.

ГЛАВА 3



Быстрый старт

Листинги данной главы
можно найти в подкаталоге start.

Несмотря на то, что PHP является универсальным языком программирования и может применяться для разработки практически любого программного обеспечения, основная его специализация — Web-разработка.

В текущей главе будут показаны наиболее типичные приемы работы с PHP. Часть конструкций (`if`, `include`) будут освещены вскользь. В последующих главах мы остановимся на них более детально.

3.1. Скрипты

Язык программирования PHP считается скриптовым языком, поэтому программы, написанные на нем, называют *скриптами*. Главное отличие традиционных программ от скриптов заключается в том, что скрипты работают только в определенной среде и используют ресурсы данной среды.

Например, скриптовый язык программирования JavaScript работает преимущественно в Web-браузерах, Visual Basic for Applications — только в среде Microsoft Office. С использованием этих языков программирования невозможно создать программу, работающую без соответствующей среды.

В случае PHP в качестве такой среды выступает Web-окружение (Web-сервер, сервер базы данных, почтовый сервер и т. п.). Именно он принимает запросы от клиента, обеспечивает их параллельное выполнение и отправку данных. PHP-скрипт получает всю информацию о запросе и выполняет запрос, а затем отправляет данные обратно серверу.

ЗАМЕЧАНИЕ

Впрочем, язык программирования PHP допускает создание программ, работающих независимо от Web-сервера, однако в такой форме он не получил сколько бы то ни было широкого распространения.