

Максим Кузнецов
Игорь Симдянов

PHP 5/6

Санкт-Петербург
«БХВ-Петербург»
2010

УДК 681.3.068+800.92РНР5/6
ББК 32.973.26–018.1
К89

Кузнецов, М. В.

К89 РНР 5/6 / М. В. Кузнецов, И. В. Симдянов. — СПб.: БХВ-Петербург, 2010. — 1024 с.: ил. — (В подлиннике)

ISBN 978-5-9775-0304-4

Рассмотрены самые последние версии языка разработки серверных сценариев РНР — 5.3 и 6.0. Подробно описан язык РНР, в том числе вопросы объектно-ориентированного программирования на РНР, обработки исключительных ситуаций, взаимодействия с MySQL и многое другое. В книге обсуждаются все вопросы, с которыми может столкнуться Web-разработчик, начиная с создания инструментария для быстрой разработки Web-приложений и последних нововведений языка программирования РНР и заканчивая вопросами безопасности и особенностями программирования клиент-серверных приложений. Книга имеет практическую направленность, т. к. содержит множество примеров, взятых из практики разработки динамических Web-сайтов.

Для программистов и Web-разработчиков

УДК 681.3.068+800.92РНР5/6
ББК 32.973.26–018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.09.09.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 82,56.

Тираж 2000 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Введение.....	1
Нововведения PHP 6	1
Путеводитель по книге	2
On-line-поддержка книги.....	4
Благодарности	4
Глава 1. Интернет	5
1.1. История развития Интернета.....	5
1.2. Принципы работы Интернета	7
1.3. Место и роль PHP в Интернете.....	12
1.3.1. Серверные технологии	13
UNIX-подобная операционная система.....	13
Web-сервер	13
Серверный язык	13
Файлы и базы данных.....	14
Электронная почта.....	15
1.3.2. Клиентские технологии.....	15
Web-браузеры, HTML	15
Каскадные таблицы стилей CSS и XML.....	16
Flash-ролики	16
FTP-клиенты.....	17
Удаленный доступ к серверу. Протокол SSH	17
Глава 2. Быстрый старт.....	18
2.1. Скрипты.....	18
2.2. Начальные и конечные теги	20
2.3. Использование точки с запятой	22
2.4. Составные выражения. Фигурные скобки	24
2.5. Комментарии	25
Глава 3. Переменные и типы данных	27
3.1. Объявление переменной. Оператор =	27
3.2. Типы данных.....	28
3.3. Целые числа	29

3.4. Вещественные числа	30
3.5. Строки	31
3.6. Кавычки.....	31
3.7. Оператор <<<.....	35
3.8. Обращение к неинициализированной переменной. Замечания (Notice)	35
3.9. Специальный тип <i>NULL</i>	37
3.10. Логический тип.....	38
3.11. Уничтожение переменной. Конструкция <i>unset()</i>	38
3.12. Проверка существования переменной. Конструкции <i>isset()</i> и <i>empty()</i>	39
3.13. Определение типа переменной	41
3.14. Неявное приведение типов	46
3.15. Явное приведение типов.....	48
3.16. Динамические переменные	52
Глава 4. Константы.....	54
4.1. Объявление константы. Функция <i>define()</i>	54
4.2. Функции для работы с константами.....	58
4.3. Константы с динамическими именами. Функция <i>constant()</i>	58
4.4. Проверка существования константы	59
4.5. Предопределенные константы	60
Глава 5. Операторы и конструкции языка.....	61
5.1. Объединение строк. Оператор "точка".....	61
5.2. Конструкция <i>echo</i> . Оператор "запятая"	62
5.3. Арифметические операторы.....	63
5.4. Поразрядные операторы	67
5.5. Операторы сравнения	72
5.6. Условный оператор <i>if</i>	75
5.7. Логические операторы.....	77
5.8. Условный оператор <i>x ? y : z</i>	84
5.9. Переключатель <i>switch</i>	85
5.10. Цикл <i>while</i>	90
5.11. Цикл <i>do ... while</i>	96
5.12. Цикл <i>for</i>	97
5.13. Включение файлов	101
5.14. Подавление вывода ошибок. Оператор @	106
5.15. Приоритет выполнения операторов	108
Глава 6. Массивы	110
6.1. Создание массива	110
6.2. Ассоциативные и индексные массивы	116
6.3. Многомерные массивы	121
6.4. Интерполяция элементов массива в строки.....	122
6.5. Конструкция <i>list()</i>	123

6.6. Обход массива	125
6.7. Цикл <i>foreach</i>	129
6.8. Сечения массива.....	131
6.9. Слияние массивов	133
6.10. Сравнение массивов.....	136
6.11. Проверка существования элементов массива.....	140
6.12. Строки как массивы	144
6.13. Количество элементов в массиве.....	145
6.14. Сумма элементов массива	147
6.15. Случайные элементы массива.....	147
6.16. Сортировка массивов	150
6.17. Добавление/удаление элементов массива.....	158
6.18. Работа с ключами массива	165
6.19. Преобразование массивов	170
6.20. Суперглобальные массивы. Массив <code>\$_SERVER</code>	177
6.20.1. Элемент <code>\$_SERVER['DOCUMENT_ROOT']</code>	178
6.20.2. Элемент <code>\$_SERVER['HTTP_ACCEPT']</code>	178
6.20.3. Элемент <code>\$_SERVER['HTTP_ACCEPT_LANGUAGE']</code>	179
6.20.4. Элемент <code>\$_SERVER['HTTP_HOST']</code>	180
6.20.5. Элемент <code>\$_SERVER['HTTP_REFERER']</code>	180
6.20.6. Элемент <code>\$_SERVER['HTTP_USER_AGENT']</code>	181
6.20.7. Элемент <code>\$_SERVER['REMOTE_ADDR']</code>	182
6.20.8. Элемент <code>\$_SERVER['SCRIPT_FILENAME']</code>	182
6.20.9. Элемент <code>\$_SERVER['SERVER_NAME']</code>	182
6.20.10. Элемент <code>\$_SERVER['REQUEST_METHOD']</code>	183
6.20.11. Элемент <code>\$_SERVER['QUERY_STRING']</code>	183
6.20.12. Элемент <code>\$_SERVER['PHP_SELF']</code>	184
6.20.13. Элемент <code>\$_SERVER['REQUEST_URI']</code>	184
Глава 7. Функции.....	185
7.1. Объявление и вызов функции	185
7.2. Параметры функции.....	188
7.3. Передача параметров по значению и ссылке	189
7.4. Необязательные параметры.....	190
7.5. Переменное количество параметров	191
7.6. Глобальные переменные.....	193
7.7. Статические переменные.....	194
7.8. Возврат массива функцией.....	195
7.9. Рекурсивные функции.....	196
7.10. Вложенные функции.....	197
7.11. Динамическое имя функции.....	198
7.12. Анонимные функции	199
7.13. Проверка существования функции.....	201
7.14. Неявное выполнение функций. Оператор <i>declare()</i>	206
7.15. Вспомогательные функции	211

Глава 8. Взаимодействие PHP с HTML	214
8.1. Передача параметров методом <i>GET</i>	214
8.2. HTML-форма и ее обработчик	218
8.3. Текстовое поле	223
8.4. Поле для приема пароля	224
8.5. Текстовая область	225
8.6. Скрытое поле	226
8.7. Флажок	227
8.8. Список	229
8.9. Переключатель	231
8.10. Загрузка файла на сервер	232
Глава 9. Строковые функции	237
9.1. Функции для работы с символами	237
9.2. Поиск в строке	243
9.3. Замена в тексте	248
9.4. Преобразование регистра	253
9.5. Работа с HTML-кодом	255
9.6. Экранирование	264
9.7. Локальные настройки (локаль)	266
9.8. Форматный вывод	273
9.9. Преобразование кодировок	282
9.10. Сравнение строк	292
9.11. Хранение данных	298
9.12. Работа с путями к файлам и каталогам	301
9.13. Объединение и разбиение строк	303
Глава 10. Регулярные выражения	313
10.1. Как изучать регулярные выражения?	313
10.2. Синтаксис регулярных выражений	314
10.3. Функции для работы с регулярными выражениями	317
10.4. Функция <i>preg_match()</i>	318
10.5. Функция <i>preg_match_all()</i>	323
10.6. Функция <i>preg_replace()</i>	326
10.7. Функция <i>preg_replace_callback()</i>	330
10.8. Функция <i>preg_split()</i>	332
10.9. Функция <i>preg_quote()</i>	334
Глава 11. Дата и время	336
11.1. Формирование даты и времени	336
11.2. Географическая привязка	342
11.3. Форматирование даты и времени	348

Глава 12. Математические функции	362
12.1. Предопределенные константы	362
12.2. Поиск максимума и минимума	363
12.3. Генерация случайных чисел	365
12.4. Преобразование значений между различными системами счисления.....	367
12.5. Округление чисел	370
12.6. Логарифмические и степенные функции	373
12.7. Тригонометрические функции	378
12.8. Информационные функции	383
12.9. Вычисления с повышенной точностью	389
Глава 13. Файлы и каталоги	392
13.1. Создание файлов	392
13.2. Манипулирование файлами	398
13.3. Чтение и запись файлов	401
13.3.1. Чтение файлов.....	404
13.3.2. Запись файлов	410
13.3.3. Обязательно ли закрывать файлы?	414
13.3.4. Дозапись файлов.....	415
13.3.5. Уменьшение размера файла	416
13.3.6. Блокировка файлов.....	417
13.3.7. Файлы CSV.....	422
13.3.8. Чтение и разбор ini-файла.....	424
13.3.9. Прямое манипулирование файловым указателем	427
13.4. Создание и работа со ссылками	430
13.5. Атрибуты файла.....	432
13.6. Права доступа	438
13.7. Состояние файловой системы	442
13.8. Каталоги	443
13.9. Архивные файлы	450
Глава 14. HTTP-заголовки	454
14.1. Функции для управления HTTP-заголовками	455
14.2. Кодировка страницы	457
14.3. HTTP-коды состояния.....	457
14.4. Список HTTP-заголовков	461
14.5. Подавление кэширования	464
Глава 15. Cookie	468
Глава 16. Сессии	473
16.1. Функции обработки сессии	473
16.2. Перегрузка механизма сессии	477

Глава 17. Управление выводом.....	481
17.1. Функции управления выводом.....	481
17.2. Размер и тип документа.....	483
17.3. Подсветка ссылок.....	486
17.4. Вложенная буферизация.....	487
17.5. Обработчики буферов.....	488
17.6. Список применяемых обработчиков.....	489
17.7. Сжатие страницы.....	491
17.8. Преобразование кодировки страницы.....	491
17.9. Преобразование адресов ссылок.....	492
Глава 18. Средства шифрования	494
18.1. Необратимое шифрование.....	494
18.2. Система регистрации и аутентификации.....	497
18.3. Обратимое шифрование.....	501
Глава 19. Сетевое взаимодействие. Сокеты и CURL.....	502
19.1. Обращение к удаленным страницам.....	502
19.2. Сокеты.....	504
19.3. Библиотека CURL.....	507
19.4. Получение HTTP-заголовков с сервера.....	515
19.5. Определение размера файла на удаленном хосте.....	519
19.6. Отправка данных методом POST.....	519
19.7. Фальсификация реферера.....	522
19.8. Фальсификация пользовательского агента.....	526
19.9. Фальсификация cookie.....	527
19.10. Работа с доменами и IP-адресами.....	530
19.11. Определение IP-адреса по сетевому адресу.....	530
19.12. Определение сетевого адреса по IP-адресу.....	531
19.13. Следование реферальному серверу.....	532
Глава 20. Взаимодействие с интерпретатором PHP.....	535
20.1. Настройки конфигурационного файла php.ini.....	535
20.1.1. Параметры языка PHP.....	536
20.1.2. Директивы безопасности.....	539
20.1.3. Настройка подсветки PHP-кода.....	545
20.1.4. Кэш файловой системы.....	547
20.1.5. Взаимодействие с клиентом.....	547
20.1.6. Ограничение ресурсов.....	548
20.1.7. Обработка ошибок.....	549
20.1.8. Обработка данных.....	552
20.1.9. Загрузка файлов.....	558
20.1.10. Сетевой доступ.....	558

20.1.11. Подключение расширений	558
20.1.12. Настройка сессии	559
20.1.13. Настройка даты и времени	561
20.2. Изменение настроек <code>php.ini</code> средствами Apache	561
20.3. Функции управления интерпретатором PHP	564

Глава 21. Объекты и классы..... 567

21.1. Введение в объектно-ориентированное программирование.....	567
21.2. Создание класса.....	568
21.3. Создание объекта	569
21.4. Инкапсуляция. Спецификаторы доступа	571
21.5. Методы класса. Член <code>\$this</code>	572
21.6. Дамп объекта	576
21.7. Специальные методы класса	577
21.8. Функции для работы с методами и классами	578
21.9. Конструктор. Метод <code>__construct()</code>	580
21.10. Параметры конструктора.....	582
21.11. Деструктор. Метод <code>__destruct()</code>	584
21.12. Автозагрузка классов. Функция <code>__autoload()</code>	586
21.13. Аксессоры. Методы <code>__set()</code> и <code>__get()</code>	586
21.14. Проверка существования члена класса. Метод <code>__isset()</code>	588
21.15. Уничтожение члена класса. Метод <code>__unset()</code>	589
21.16. Динамические методы. Метод <code>__call()</code>	591
21.17. Интерполяция объекта. Метод <code>__toString()</code>	593
21.18. Экспорт объектов. Метод <code>__set_state()</code>	595
21.19. Статические члены класса.....	600
21.20. Статические методы класса.....	603
21.21. Константы класса	603
21.22. Предопределенные константы	605
21.23. Клонирование объекта.....	607
21.24. Управление процессом клонирования. Метод <code>__clone()</code>	608
21.25. Сериализация объектов	610
21.26. Управление сериализацией. Методы <code>__sleep()</code> и <code>__wakeup()</code>	612

Глава 22. Наследование 621

22.1. Наследование.....	621
22.2. Спецификаторы доступа и наследование	624
22.3. Перегрузка методов.....	627
22.4. Полиморфизм	628
22.5. Абстрактные классы	630
22.6. Абстрактные методы.....	631
22.7. <i>Final</i> -методы класса	632
22.8. <i>Final</i> -классы	634

Глава 23. Интерфейсы	635
23.1. Создание интерфейса	635
23.2. Интерфейсы и наследование классов	637
23.3. Реализация нескольких интерфейсов	639
23.4. Проверка существования интерфейса	640
23.5. Наследование интерфейсов	641
23.6. Реализует ли объект интерфейс?	643
Глава 24. Обработка ошибок и исключительных ситуаций	647
24.1. Обработка ошибок	647
24.2. Обработка ошибок и исключения.....	660
24.2.1. Синтаксис исключений	660
24.2.2. Интерфейс класса <i>Exception</i>	662
24.2.3. Генерация исключений в функциях.....	666
24.2.4. Стек обработки исключительной ситуации	669
24.2.5. Генерация исключений в классах	671
24.2.6. Генерация исключений в иерархиях классов.....	675
24.2.7. Использование объекта класса <i>Exception</i> в строковом контексте	678
24.2.8. Создание собственных исключений	679
24.2.9. Создание новых типов исключений	683
24.2.10. Перехват исключений производных классов	685
24.2.11. Что происходит, когда исключения не перехватываются?	687
24.2.12. Вложенные контролируемые блоки	689
24.2.13. Повторная генерация исключений.....	691
Глава 25. Отражения	694
25.1. Иерархия классов отражения	694
25.2. Отражение функции. Класс <i>ReflectionFunction</i>	695
25.3. Отражение параметра функции. Класс <i>ReflectionParameter</i>	700
25.4. Отражение класса. Класс <i>ReflectionClass</i>	702
25.5. Отражение объекта. Класс <i>ReflectionObject</i>	709
25.6. Отражение метода класса. Класс <i>ReflectionMethod</i>	710
25.7. Отражение члена класса. Класс <i>ReflectionProperty</i>	712
25.8. Исключения механизма отражения	715
25.9. Отражение расширения. Класс <i>ReflectionExtension</i>	716
25.10. Вспомогательный класс <i>Reflection</i>	719
Глава 26. Предопределенные классы	721
26.1. Класс <i>Dir</i>	721
26.2. Классы времени	723
26.3. Библиотека SPL	730
26.3.1. Итераторы.....	731
26.3.2. Интерфейс <i>Iterator</i>	734
26.3.3. Класс <i>DirectoryIterator</i>	736

26.3.4. Класс <i>FilterIterator</i>	738
26.3.5. Класс <i>LimitIterator</i>	739
26.3.6. Рекурсивные итераторы	740
Глава 27. Работа с СУБД MySQL.....	742
27.1. Введение в СУБД и SQL.....	743
27.2. Первичные ключи.....	746
27.3. Создание и удаление базы данных	747
27.4. Выбор базы данных.....	749
27.5. Типы данных.....	751
27.6. Создание и удаление таблиц	756
27.7. Вставка числовых значений в таблицу.....	763
27.8. Вставка строковых значений в таблицу	764
27.9. Вставка календарных значений	766
27.10. Вставка уникальных значений	769
27.11. Механизм <i>AUTO_INCREMENT</i>	769
27.12. Многострочный оператор <i>INSERT</i>	770
27.13. Удаление данных.....	771
27.14. Обновление записей.....	772
27.15. Выборка данных	775
27.16. Условная выборка	776
27.17. Псевдонимы столбцов	782
27.18. Сортировка записей	783
27.19. Вывод записей в случайном порядке	786
27.20. Ограничение выборки.....	786
27.21. Вывод уникальных значений	787
27.22. Объединение таблиц	789
27.23. Функции MySQL	791
27.23.1. Математические функции.....	791
Вычисление площади треугольников	793
Округление результатов вычисления.....	795
27.23.2. Функции даты и времени	796
Форматирование календарных значений	801
Вычисление возраста человека.....	802
Преобразование даты в UNIXSTAMP-формат	804
27.23.3. Строковые функции	805
Изменение кодировки строки.....	809
Первые несколько символов строки	810
Извлечение инициалов	810
Изменение регистра строки	811
27.23.4. Функции шифрования.....	812
Обратимое шифрование.....	814
Необратимое шифрование	815
27.23.5. Агрегатные функции	816
Среднее значение.....	817
Сортировка агрегатных значений	820

Подсчет количества записей в таблице	820
Объединение значений группы	823
Поиск минимального и максимального значений	825
Сумма столбца	826
27.23.6. Разное.....	827
Преобразование IP-адреса.....	828
Глава 28. Взаимодействие MySQL и PHP	830
28.1. Библиотека <code>php_mysql</code>	830
28.1.1. Установка соединения с базой данных	830
28.1.2. Выбор базы данных	833
28.1.3. Выполнение SQL-запросов	835
28.1.4. Получение результатов запроса	836
28.1.5. Количество строк в таблице	844
28.1.6. Экранирование данных. SQL-инъекции.....	846
28.2. Библиотека <code>php_mysqli</code>	853
28.2.1. Создание базы данных	860
28.2.2. Создание и заполнение таблицы	861
28.2.3. Заполнение связанных таблиц.....	862
28.2.4. Вывод данных	865
28.2.5. Повторное чтение результирующей таблицы.....	867
28.2.6. Количество строк в таблице	868
28.2.7. Удаление данных	869
28.2.8. Сортировка	871
28.2.9. Параметризация SQL-запросов	873
Глава 29. FTP-менеджер.....	875
29.1. Функции для работы с FTP-сервером.....	875
29.2. Какой объем жесткого диска занимает сайт?	879
29.3. Перенос сайта с одного хоста на другой.....	881
Глава 30. Электронная почта	885
30.1. Отправка почтового сообщения.....	885
30.2. Рассылка писем.....	887
30.3. Отправка писем с вложением.....	888
30.4. Отправка писем со встроенными изображениями	891
Глава 31. Динамические изображения. Библиотека GDLib	894
31.1. Информационные функции	894
31.1.1. Текущая версия библиотеки GDLib.....	895
31.1.2. Формат файла.....	897
31.1.3. Размер файла	898
31.1.4. Получение MIME-типа файла	899
31.2. Функции создания изображений.....	901

31.3. Функции сохранения и вывода изображений.....	903
31.4. Функции преобразования изображений.....	905
31.4.1. Создание уменьшенной копии изображения.....	906
31.4.2. Поворот изображения.....	908
31.5. Функции для работы с цветом.....	909
31.5.1. Заливка изображения цветом.....	911
31.5.2. Получение цвета заданного пиксела.....	913
31.5.3. Уменьшение количества цветов в изображении.....	914
31.5.4. Замена одного цвета другим.....	915
31.6. Функции рисования.....	916
31.6.1. Кривая Безье.....	918
31.6.2. Построение гистограммы средствами GDLib.....	922
31.6.3. Построение круговой диаграммы средствами GDLib.....	924
31.7. Функции настройки рисования.....	925
31.7.1. Пунктирная линия.....	926
31.8. Функции для работы с текстом.....	928
31.8.1. Защитное изображение для HTML-формы.....	930

Заключение.....	935
------------------------	------------

ПРИЛОЖЕНИЯ.....	937
------------------------	------------

Приложение 1. Установка и настройка PHP, Web-сервера Apache и MySQL-сервера.....	939
---	------------

П1.1. Где взять дистрибутивы?.....	939
П1.1.1. Дистрибутив PHP.....	940
П1.1.2. Дистрибутив Apache.....	940
П1.1.3. Дистрибутив MySQL.....	941
П1.2. Установка Web-сервера Apache под Windows.....	942
П1.3. Установка Web-сервера Apache под Linux.....	944
П1.4. Настройка виртуальных хостов.....	945
П1.5. Настройка кодировки по умолчанию.....	949
П1.6. Управление запуском и остановкой Web-сервера Apache.....	949
П1.7. Управление Apache из командной строки.....	950
П1.8. Установка PHP под Windows.....	951
П1.8.1. Установка PHP в качестве модуля.....	951
П1.8.2. Установка PHP как CGI-приложения.....	952
П1.9. Установка PHP под Linux.....	954

Приложение 2. Установка MySQL.....	955
---	------------

П2.1. Установка MySQL под Windows.....	955
П2.1.1. Процесс установки.....	955
П2.1.2. Постинсталляционная настройка.....	960
П2.1.3. Проверка работоспособности MySQL.....	966

П2.2. Установка MySQL под Linux.....	968
П2.3. Конфигурационный файл	971
П2.4. Утилита mysql	974
П2.4.1. Командная строка	974
П2.4.2. Работа с утилитой mysql в диалоговом режиме.....	978
П2.5. Перенос баз данных с одного сервера на другой	982
П2.5.1. Копирование бинарных файлов	983
П2.5.2. Создание SQL-дампа	984
Рекомендуемая литература	987
HTML, XML, CSS, JavaScript и Flash	989
PHP и Perl	991
СУБД MySQL	993
Интернет и Web-сервер Apache	994
Регулярные выражения	995
UNIX-подобные операционные системы.....	995
Методология программирования.....	997
Предметный указатель.....	999

Введение

PHP является молодым и динамично развивающимся языком программирования, который используется главным образом для создания самых разнообразных Web-приложений. Быстрое развитие языка, с одной стороны, предполагает регулярное введение новых конструкций, функций, библиотек, директив, а с другой стороны, — изъятие старых конструкций, использование которых оказалось неудобным или провоцировало создание небезопасного кода. Стадию динамичного развития проходит любой язык программирования, определить ее очень просто по быстрой смене версий и количеству нововведений в этих версиях. Со временем интервал между версиями увеличивается, каждая новая версия содержит все меньше и меньше революционных нововведений. Современный PHP пока находится в стадии быстрого развития, однако в ближайшие несколько лет прогнозируется переход к плавному изменению, характерному для состоявшихся языков.

Анонсируя PHP 6, разработчики заранее объявили о нововведениях, которые будут в новой версии. Однако затем ими было принято решение не создавать дистрибутив с нуля, а постепенно вносить изменения в текущую версию PHP 5. Если раньше, для того чтобы попробовать новинки языка, требовалось загрузить нестабильный дистрибутив, предназначенный для разработчиков, то теперь особенности PHP 6 постепенно и плавно появляются в версии PHP 5. Таким образом, к тому моменту, когда все обновления будут внедрены в язык, разработчикам останется только поменять цифру. Поэтому нами было принято решение о создании руководства по языку PHP с учетом всех последних нововведений.

Эта книга будет интересна как начинающим разработчикам, которым необходимо последовательное изложение языка, так и опытным программистам, желающим познакомиться с нововведениями языка, появившимися в последнее время.

Нововведения PHP 6

Не все нововведения языка реализованы на данный момент, — к примеру, нет сечений массивов и строк, отсутствует (в стабильных версиях) поддержка Unicode на уровне ядра, в расширении для работы с графикой не поддерживается работа с анимированными GIF-файлами. Список этот можно еще продолжить, однако интереснее отметить то, что уже реализовано:

- исправлены многочисленные непоследовательности объектно-ориентированной модели;

- значение `E_ALL` директивы `error_reporting` конфигурационного файла `php.ini` теперь включает `E_STRICT` (см. разд. 3.8);
- средний параметр `y` в условном операторе `x ? y : z`, начиная с PHP 6.0, не является обязательным (см. разд. 5.8);
- ускорена работа оператора `@` (см. главу 5);
- функциональность цикла `foreach` расширена для работы с многомерными массивами;
- в операторе `break` удалена поддержка динамических меток;
- регулярные выражения POSIX исключены из ядра PHP и доступны лишь как расширение;
- введены новые predefined массивы для работы с датой и временем;
- изменен синтаксис множества функций (главным образом, за счет увеличения числа необязательных параметров).

Путеводитель по книге

Книга содержит введение, 31 главу, заключение и два приложения. В конце книги вы найдете обзор литературы в области Web-разработки с подробными комментариями авторов.

Глава 1 посвящена краткой истории развития Интернета, а также места и роли языка PHP в Web-технологиях, благодаря которым стало возможным существование всемирной сети.

Глава 2 содержит описание простейших базовых конструкций языка и служит введением для тех читателей, для которых PHP является первым языком программирования.

Глава 3 посвящена переменным и подробно освещает жизненный цикл переменных: создание, уничтожение, отличие типов переменных друг от друга, явное и неявное преобразование типов переменных.

Глава 4 посвящена пользовательским и predefined константам, а также функциям, обслуживающим константы.

Глава 5 описывает операторы и конструкции языка, подробно освещая как классические C-подобные операторы: условные, циклические, логические, поразрядные, арифметические, так и специфические для PHP операторы: "точка", "запятая", оператор подавления ошибок, конструкции включения файлов.

Глава 6 посвящена описанию массивов, подробно рассматривает как типы массивов, порядок работы с ними, вспомогательные функции. В главе также рассматривается суперглобальный массив `$_SERVER`, позволяющий взаимодействовать PHP-скрипту с окружением (операционной системой и Web-сервером).

Глава 7 описывает функции, создание собственных функций, их обслуживание, динамические, анонимные функции, функции с переменным количеством параметров и т. п.

Глава 8 посвящена взаимодействию PHP и языка разметки HTML. Львиная доля PHP-скриптов занимается обслуживанием многочисленных форм. В главе описывается син-

таксис элементов формы и типичные случаи обработки на PHP поступающих из форм данных. В этой же главе подробно описываются суперглобальные массивы `$_GET` и `$_POST`.

Глава 9 описывает строковые функции PHP: поиск и замену в строках, преобразование регистра и кодировки, форматный вывод и т. п.

Глава 10 посвящена регулярным выражениям — специализированному мини-языку, позволяющему эффективно работать со строками (осуществлять поиск, подстановку, удаление подстрок).

Глава 11 посвящена функциям, работающим с датой и временем.

Глава 12 описывает математические функции, затрагивая тригонометрические, логарифмические функции, функции общего назначения, а также функции, предназначенные для вычислений с повышенной точностью.

Глава 13 посвящена файлам и каталогам и включает подробные сведения о файловой системе, а также функциях PHP, обслуживающих файловую систему.

Глава 14 посвящена протоколу HTTP, являющемуся основой Web-среды. Язык PHP спроектирован таким образом, чтобы по возможности оградить разработчика от необходимости вникать в низкоуровневые реализации Сети. Однако эффективные Web-приложения без знания протокола HTTP освоить невозможно. В любом случае, необходимо четко понимать, как PHP реализуют те или иные сетевые возможности.

Глава 15 описывает механизм cookie, предназначенный для сеансовой работы клиентов (не поддерживаемой на уровне протокола HTTP).

Глава 16 посвящена сессиям, также предназначенным для реализации сеансовой работы.

Глава 17 описывает процесс буферизации вывода. Все, что выводится PHP-скриптом, немедленно отправляется клиенту и зачастую трудно выполнить какие-то преобразования над уже сформированным HTML-кодом. Данная глава описывает решение этой проблемы путем создания буферов вывода и манипулирования ими.

Глава 18 посвящена средствам шифрования, предоставляемым PHP.

Глава 19 описывает сокет и библиотеку CURL, позволяющие скрипту самому выступать в роли клиента, обращаясь к удаленным серверам.

Глава 20 описывает настройки PHP-интерпретатора, а также способы управления ими как на уровне Web-сервера, так и отдельного скрипта.

Глава 21 посвящена введению в объектно-ориентированный подход, позволяющий создавать повторно-используемый код и собственные мини-языки, работающие в предметной области разработки.

Глава 22 подробно описывает наследование в объектно-ориентированном подходе PHP.

Глава 23 описывает интерфейсы в объектно-ориентированном подходе PHP.

Глава 24 посвящена обработке ошибок в PHP-скриптах, как при помощи специализированных функций, характерных для структурного подхода, так и при помощи исключений, характерных для объектно-ориентированного подхода.

Глава 25 посвящена механизму отражений, предоставляющему разработчику возможность исследования как пользовательских, так и predefined классов. Кроме

этого, отражения позволяют автоматически генерировать документацию иерархии классов по схеме javadoc, применяемой в технологии Java.

Глава 26 описывает predefined классы PHP, такие как класс `Dir` доступа к каталогам, классы даты и времени, классы библиотеки SPL.

Глава 27 посвящена базе данных MySQL — наиболее популярной базе данных, используемой совместно с PHP.

Глава 28 описывает взаимодействие PHP и MySQL.

Глава 29 посвящена FTP-протоколу, при помощи которого осуществляется взаимодействие с файловой системой удаленного сервера.

Глава 30 описывает процесс отправки электронных писем из PHP-скрипта, включая письма, содержащие вложения и встроенные изображения.

Глава 31 посвящена созданию динамических изображений при помощи библиотеки GDLib.

On-line-поддержка книги

Разнообразные дополнительные материалы вы можете найти на сайтах IT-студии Soft-Time, руководителями которой являются авторы книги:

- ☐ <http://www.softtime.ru> — головной сайт Студии;
- ☐ <http://www.softtime.org> — различные проекты Студии;
- ☐ <http://www.softtime.biz> — услуги, оказываемые нашей Студией;
- ☐ <http://www.softtime.mobi> — вариант портала для мобильных устройств.

Обсудить вопросы, которые могут возникать по мере чтения книги, можно на форуме авторов: <http://www.softtime.ru/forum/>.

Благодарности

Авторы выражают огромную благодарность сотрудникам издательства "БХВ-Петербург", благодаря которым наша рукопись увидела свет.

Мы также безмерно признательны посетителям наших форумов за интересные вопросы и конструктивное обсуждение.



Глава 1

Интернет

По данным за 2007 год количество пользователей глобальной сети Интернет достигло 1 миллиарда человек. Интернет является одним из самых грандиозных инженерных достижений человечества, изменившим как повседневную жизнь людей, так и облик современной цивилизации. В данной главе мы кратко рассмотрим историю развития Интернета и его устройство.

1.1. История развития Интернета

Прототипом Интернета послужила первая глобальная сеть ARPANET, разработка которой финансировалась Управлением перспективного планирования научно-исследовательских работ (Advanced Research Projects Agency, ARPA) Министерства обороны Соединенных Штатов Америки (США).

Цель проекта была двойной. В идеале, необходимо было создать распределенную глобальную компьютерную сеть, способную продолжать функционировать после ядерной войны. Вывод из строя произвольного количества узлов сети не должен был приводить к выходу из строя сети в целом.

Более приземленная цель состояла в создании сети для обмена файлами и электронной почтой между университетскими учеными, сотрудниками военных ведомств и подрядчиками Министерства обороны США.

Так или иначе, осенью 1968 года ARPA заключило контракт на разработку сети ARPANET с тремя сотрудниками Кембриджского университета — Болтом (Bolt), Беранеком (Beranek) и Ньюменом (Newman), и уже в сентябре 1969 года первый участок сети был запущен в опытную эксплуатацию.

В 1975 году управление сетью было выведено из-под контроля ARPA и поручено управлению связи Министерства обороны США, которое сделало ARPANET частью собственной сети.

Одной из главных особенностей финансирования научных работ государством является то, что все разработки ведутся на деньги налогоплательщиков и не являются интеллектуальной собственностью одной из корпораций, а автоматически становятся достоянием общественности. Любой разработчик может создать программу или аппаратное обеспечение для обмена информацией через сеть. Это с одной стороны привлекло множество энтузиастов, а с другой стороны определило в дальнейшем широкое распространение сетевых технологий по всему миру.

Несмотря на то, что сеть была первоначально построена на выделенных линиях, агентство ARPA финансировало исследования технологий передачи пакетов данных по радиоканалу и каналам спутниковой связи.

В связи выделяют два способа передачи данных: обмен информацией через выделенные линии связи и обмен с коммутацией пакетов. Выделенные линии после установки соединения остаются занятыми, даже если абоненты не обмениваются информацией (так построены телефонные линии). При коммутации пакетов, данные разбиваются на части со стороны одного абонента и собираются из разрозненных частей со стороны второго абонента. В результате по одной линии связи можно передавать данные одновременно от нескольких абонентов. Именно так функционирует современный Интернет, а также IP-телефония (которая по сути использует интернет-соединения для передачи голоса). Коммутация пакетов более сложна в реализации по сравнению с выделенными каналами, однако позволяет более эффективно использовать каналы связи.

Переход на линии связи с коммутацией пакетов в сети ARPANET был начат в 1980 году в связи с введением протоколов семейства TCP/IP и закончен в 1983 году.

Так как сеть разрасталась и в ее работе участвовало все больше гражданских лиц, в 1983 году Министерство обороны разделило ARPANET на две связанные сети: гражданский сектор, предназначенный для исследований и обмена открытой информацией (по-прежнему носящий название ARPANET), и сеть для военных целей (получившая название MILNET).

ЗАМЕЧАНИЕ

Управление сетями было организовано таким образом, что в любой момент их можно было разъединить. Когда в ноябре 1988 года произошло заражение гражданской части сети первым червем, созданным Моррисом, такая возможность помогла предотвратить попадание червя в MILNET. Данная ситуация даже немного обыгрывается в фильме "Терминатор. Восстание машин".

Чтобы заинтересовать университеты в использовании новых протоколов TCP/IP, Министерство обороны стало продавать их программную реализацию по низкой цене, а также профинансировало разработку реализации протоколов TCP/IP для BSD UNIX (популярного варианта UNIX-подобной операционной системы, разработанной в Калифорнийском университете Беркли). В результате протоколы TCP/IP стали доступными более чем в 90% университетских компьютерных лабораториях и факультетах. Сеть ARPANET стала одним из главных магистральных каналов, к которому стали подключаться другие сети, образуя тем самым Всемирную глобальную сеть — Интернет (Internet).

Так как все больше университетов соединяли свои вычислительные возможности в локальные сети, а также подключались к ARPANET, Национальный научный фонд США (National Science Foundation или NSF) начал финансирование проекта сети CSNET (Computer Science NETwork), которая должна была связать все научные компьютерные лаборатории. В 1985 году сетью CSNET были соединены шесть суперкомпьютеров, принадлежавших Национальному научному фонду. В 1986 году NSF выделяет средства на создание новой глобальной магистральной сети NSFNET, которая связала все суперкомпьютерные центры США и объединила их с ARPANET. В том же 1986 году NSF финансирует множество проектов региональных сетей, каждая из которых связала основные научно-исследовательские учреждения в своем регионе.

ЗАМЕЧАНИЕ

Сеть CSFNET состояла из магистрального канала высокой пропускной способности, охватывающего всю территорию США, к которому подключались местные локальные сети, охватывающие университеты и государственные учреждения.

С этого момента к Интернету стало подключаться множество частных сетей в США и Западной Европе. Темпы роста Интернета составляли более 1000% в год и поддерживались не только государственными учреждениями, к Всемирной сети стало подключаться все больше и больше частных компаний.

В 1991 году фонд NSF и другие государственные учреждения США столкнулись с фактом, что Интернет де-факто перестал быть университетской и научной сетью. Магистральный канал (с пропускной способностью 1,5 Мбит/с), к которому подключались все новые и новые сети, был перегружен. Поэтому с 1991 года правительство США приняло решение позволить приватизировать магистральный канал и арендовать его у частной компании, которая бы самостоятельно заботилась об увеличении его пропускной способности.

В декабре 1991 года три корпорации — IBM, MERIT и MCI — учредили некоммерческую организацию Advanced Network and Services (ANS), которая в 1993 году ввела в строй новый магистральный канал с пропускной способностью 45 Мбит/с. В 1995 году пропускная способность магистрального канала была еще раз увеличена до 155 Мбит/с.

С 1995 года Интернет широко используется в коммерческих целях, некоторые корпорации (например, AT&T и MCI) построили собственные магистральные каналы, которые обмениваются трафиком на условиях партнерских соглашений.

ЗАМЕЧАНИЕ

Как правило, компания платит другой компании в том случае, если входящий трафик у ее каналов выше, чем исходящий — разница оплачивается по гигабайтно. Именно этим определяется частое требование для серверов в соотношениях входящего и исходящего трафиков, зарубежного и российского, входящего в региональное кольцо и не входящего.

Современные магистральные каналы обеспечиваются пропускной способностью в десятки гигабит в секунду. В Российской Федерации имеется несколько магистральных каналов. В качестве примера можно назвать ТрансТелеКом, которым владеет компания "Российские железные дороги". В 2007 году компания владела магистральным каналом протяженностью более 50 тыс. км, обеспечивающим пропускную способность до 50 Гбит/с.

В настоящий момент в США функционирует и развивается исследовательский проект Internet2, позволяющий достигать пропускной способности 100 Гбит/с.

1.2. Принципы работы Интернета

Сама сеть построена на множестве разнородных аппаратных и программных средств, соединенных каналами связи совершенно разной природы: начиная оптоволоконными и медными линиями связи, заканчивая беспроводными и спутниковыми каналами. Объединение таких разнородных элементов стало возможным благодаря применению

протоколов связи — сеть может быть устроена произвольным образом, однако обмениваться с другими сетями она должна пакетами определенного формата.

Протокол — это набор синтаксических и семантических правил, использующихся при обмене данными между двумя компьютерами. В нем оговаривается формат блоков сообщений, описывается реакция компьютера на получение определенного типа сообщений и указываются способы обработки ошибок и других исключительных ситуаций.

Под управлением какой бы операционной системы не работали компьютеры, кто бы ни производил комплектующие, сетевые платы, маршрутизаторы и т. п., если система выполняет предписания протокола, она может стать участником сети.

Передаваемые файлы и сообщения перед отправкой разбиваются на части — *пакеты*. Пакеты от нескольких машин передаются вперемешку от одной сети к другой. Получатель на другом конце соединения собирает разрозненные куски в одно целое.

Для того чтобы обеспечить взаимодействие различных сетей и каналов связи, используется несколько протоколов, которые работают на разных уровнях (рис. 1.1).

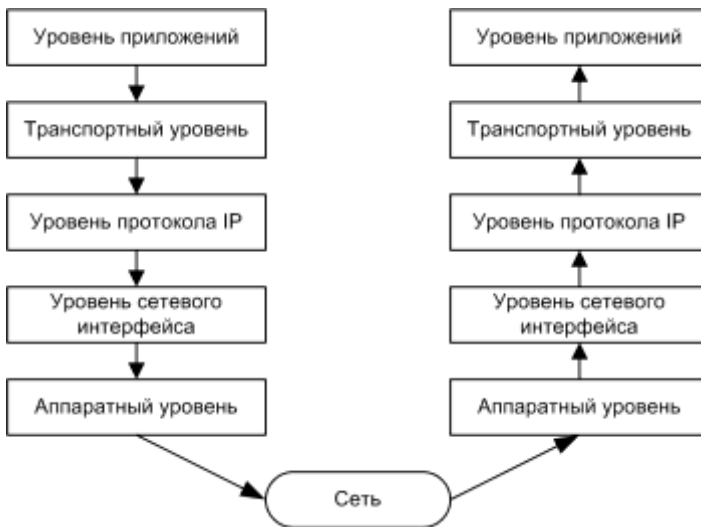


Рис. 1.1. Уровни протоколов Интернета

На *аппаратном уровне* взаимодействие определяется конкретными реализациями сетевого оборудования: это может быть оптоволокно, спутник, медный кабель, реализующий соединение через Ethernet. Задача этого уровня — передавать *сетевые фреймы*, формат которых определяется конкретным оборудованием.

На *уровне сетевого интерфейса* (который иногда называют *канальным уровнем*) протоколы обеспечивают работу сетевого оборудования, в частности платы сетевого интерфейса (реализуется как драйвер устройства). Они принимают *IP-дейтаграммы* и передают их по физической сети.

На *уровне протокола IP* обеспечивается взаимодействие двух узлов сети. Каждый из узлов обозначается уникальным IP-адресом, что позволяет обмениваться IP-дейтаграммами, содержащими в себе пакеты транспортного протокола.

На *транспортном уровне* обеспечивается передача данных между программами. Здесь используются два вида протоколов: UDP, обеспечивающий доставку пакетов, не требующую подтверждения их доставки, и TCP, обеспечивающий гарантированную доставку пакетов. Протокол UDP используется там, где потеря нескольких пакетов не имеет значения (служба точного времени, online-вещание и т. п.). Протокол TCP применяется в том случае, если ни один байт пропасть не должен (обмен файлами, почтовыми сообщениями и т. п.).

Программы, предназначенные для пользователей, работают на *уровне приложений*. Среди наиболее популярных служб прикладного уровня выделяют:

- Web — служба предоставляет доступ к связанным гиперссылками Web-сайтам, содержащим страницы с текстовой информацией и графикой. Для передачи данных в Web используется протокол HTTP;
- электронная почта — программы электронной почты позволяют рассылать копии текстовых сообщений одному или нескольким клиентам. К текстовому сообщению может быть прикреплен один или более файлов произвольного формата. Для работы с электронной почтой используются протоколы SMTP и IMAP;
- пересылка файлов — служба позволяет обмениваться файлами. Для работы с этой службой применяется протокол FTP;
- удаленный доступ к компьютерам — служба позволяет подключиться к удаленной машине и установить с ней интерактивный сеанс связи. Для установки удаленного доступа в основном используется протокол SSH;
- мгновенный обмен сообщениями — позволяет двум машинам устанавливать соединение и обмениваться короткими сообщениями. В основном используется протокол ICQ.

Здесь приведены лишь наиболее популярные службы и протоколы, на самом деле прикладной уровень содержит огромное количество протоколов и служб, начиная с протоколов электронной коммерции, заканчивая специализированными протоколами для обновления программного обеспечения через Интернет.

Таким образом, данные перед пересылкой не только разбиваются на части, но и на каждом уровне упаковываются в соответствующий формат. Для передачи файлов и данных, отправляемых на уровне приложений, открывается соединение. Файл передается на транспортный уровень и разбивается на пакеты (если речь идет о протоколе TCP), далее на уровне протокола IP пакеты могут разбиваться еще раз (или не разбиваться) и упаковываться в IP-дейтаграммы. IP-дейтаграммы, в свою очередь, опять разбиваются и упаковываются в сетевые фреймы. Таким образом, данные на каждом из уровней упаковываются по принципу матрешки (рис. 1.2).

При получении данные извлекаются в обратном порядке и объединяются для получения конечного результата.

ЗАМЕЧАНИЕ

На момент написания книги, производилась модернизация корневых DNS-серверов, позволяющая наряду с протоколом версии IPv4 использовать протокол версии IPv6, в котором вместо четырех чисел XXX.YYY.ZZZ.WWW будет использоваться шесть чисел SSS.TTT.XXX.YYY.ZZZ.WWW. Такое расширение необходимо в связи с тем, что уже сейчас ощущается нехватка адресного пространства.

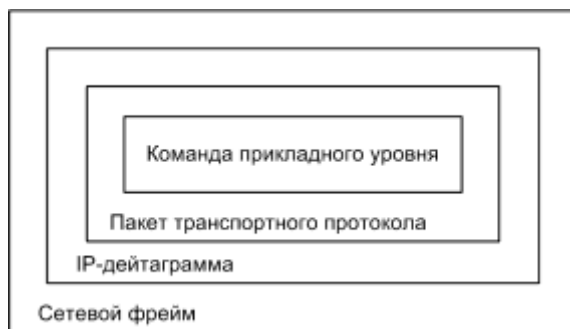


Рис. 1.2. Схема упаковки данных в семействе протоколов TCP/IP

Для того чтобы различать компьютеры в сети, каждому из них назначается IP-адрес, представляющий собой четыре числа (от 0 до 255), например, 127.0.0.1. Использовать IP-адрес для доступа к узлам сети неудобно, т. к. люди запоминают числа хуже названий. Кроме того, для обслуживания каждого сайта потребовался бы выделенный сервер и персональный IP-адрес. Так как сайтов очень много, а серверы достигли мощностей, когда они способны обслуживать сотни, а то и тысячи сайтов, такая ситуация является неприемлемой. В связи с этим в Интернете была введена служба доменных имен DNS, позволяющая установить соответствие между IP-адресом и *доменным именем*.

Например, при вводе в браузер доменного имени **mail.softtime.ru** сначала идет обращение не к серверу с сайтом, а к DNS-серверу, возвращающему IP-адрес для данного доменного имени, при помощи которого можно найти сервер в Интернете.

ЗАМЕЧАНИЕ

Назначение сайтам доменных имен позволяет безболезненно менять IP-адреса, а также располагать сайт на нескольких независимых серверах, находящихся в разных точках земного шара (что обеспечивает более быстрый доступ к сайту для участников сети).

Доменное имя представляет собой названия, разделенные точками. В доменном имени **mail.softtime.ru** часть **ru** выступает в качестве домена первого уровня, **softtime.ru** — в качестве домена второго уровня, а **mail.softtime.ru** в качестве домена третьего уровня и т. д. Домены первого уровня являются зарезервированными, часть из них предназначена для национальных сетей, например, **ru** предназначены для сайтов российского сегмента Интернета, **ua** — для украинского, **by** — белорусского и т. д. Часть доменов первого уровня предназначена для сайтов, посвященных определенному виду деятельности. Например, **com** — коммерческие компании, **org** — организации, **net** — сайты, предназначенные для развития Интернета, **gov** — правительственные учреждения, **travel** — сайты, посвященные туризму, и т. п.

ЗАМЕЧАНИЕ

Ценовая политика и порядок оформления доменного имени второго уровня определяется отдельно для каждой из зон. Если для регистрации сайтов в зонах **com**, **org** и **net** потребуются только ваши контактные данные, то для регистрации в зоне **ru** регистратор может потребовать паспортные данные. Следует учитывать, что доменные

имена второго уровня не продаются, а сдаются в аренду, если по истечении срока аренды доменное имя не будет оплачено, оно будет доступно для повторной регистрации (спустя 70 дней после окончания срока).

В каждой из зон можно зарегистрировать доменное имя второго уровня, а для доменов второго уровня можно зарегистрировать домены третьего уровня. При получении IP-адреса по доменному имени, корневому DNS-серверу отправляется домен первого уровня, который возвращает IP-адрес DNS-сервера обслуживающего эту зону, следующий DNS-сервер возвращает либо IP-адрес конечного сервера (если это домен второго уровня), либо следующий промежуточный DNS-сервер (если это домен третьего и более высокого уровня). Процедура повторяется до тех пор, пока не будет получен IP-адрес, соответствующий доменному имени.

Современные операционные системы поддерживают многозадачный режим, допускающий одновременное выполнение нескольких программ. Таким образом, данные через сеть получает процесс, запущенный на узле сети с определенным IP-адресом. Однако процессов, нуждающихся в сетевом взаимодействии, может быть запущено несколько, кроме того, они создаются и уничтожаются динамически. В связи с этим при создании сетевых соединений решено было ориентироваться не на процессы, а на функции, которые они реализуют. То есть клиент должен иметь возможность подключаться к серверу, не выясняя, в каком именно процессе реализованы функции сервера. В связи с этим были введены так называемые стандартные *порты* — целые положительные числа, за которыми закрепляется тот или иной протокол. Для протокола HTTP используется 80-й порт. Тот факт, что порт является стандартным, позволяет не указывать его при обращении к сайту, однако он может быть указан явно, например, **http://www.softtime.ru:80/index.php?id_forum=1** вместо **http://www.softtime.ru/forum/index.php?id_forum=1**. Более того, указание порта является обязательным, если сервер использует нестандартный порт, например, 8080 вместо 80. Для многих служб Интернета используются стандартные порты, например, протокол FTP использует порты 20 и 21 (один для передачи команд, другой для передачи данных), служба управления удаленным компьютером SSH закрепляет за собой порт 22, протокол электронной почты SMTP — порт 25, сервер доменных имен DNS — порт 53 и т. д.

Компьютеры в Интернете не равнозначные, часть из них используется как серверы, предназначенные для хранения данных, другие — для доступа к этим данным, третьи обеспечивают пересылку информации между сетями и т. д.

При работе с РНР мы будем взаимодействовать только с протоколами *прикладного уровня* (практически не вмешиваясь в работу более низкоуровневых протоколов), поэтому из всего многообразия компьютеров в Интернете нас будут интересовать серверы и клиентские машины.

Нас в первую очередь будут интересовать даже не компьютеры, а программы, установленные на них. *Сервер*, предоставляющий доступ к данным для клиентов, представляет собой сложную ресурсоемкую программу. Под *клиентом* обычно понимается программа, позволяющая получить доступ к серверу. Например, в качестве клиента Web-сервера выступает браузер, в качестве клиента почтового сервера — почтовый агент и т. п. Сами серверы также могут выступать в качестве клиентов, например, Web-сервер может выступать в качестве клиента базы данных, программа на РНР может выступать в качестве клиента другого Web-сервера.

1.3. Место и роль PHP в Интернете

В предыдущем разделе упоминалось, что PHP работает на прикладном уровне. Это означает, что с его помощью можно разрабатывать любые программы, использующие протоколы прикладного уровня, будь то Web-приложения, программы для отправки или получения почтовых сообщений, взаимодействие с FTP-сервером и т. п.

Web-программирование — это технология, включающая в себя множество компонентов, одним из которых выступает PHP. Прежде чем приступить к его подробному изучению, определим его место и роль в процессе обмена данными между клиентом и сервером. На рис. 1.3 представлена схема, демонстрирующая место и время действия компонентов типичной среды Web-разработки.

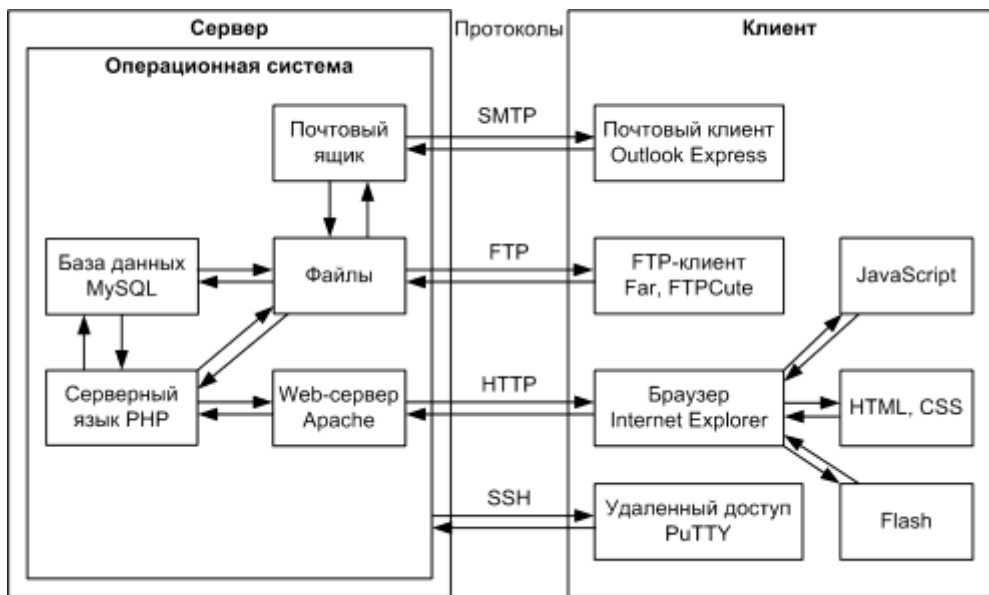


Рис. 1.3. Клиент-серверное взаимодействие

Как видно из схемы, помимо PHP в работе Web-приложений принимает участие множество дополнительных программ и технологий. Все они в той или иной степени будут затрагиваться в данной книге, т. к. PHP — всего лишь один из компонентов Web-технологии; для освоения всего процесса разработки Web-приложения необходимо понимать принципы работы всех компонентов, участвующих в схеме, представленной на рис. 1.3.

Схему взаимодействия клиента и сервера можно условно поделить на три части:

- серверные технологии — эти компоненты работают на сервере, вы не сможете запускать и использовать их на стороне клиента, если только последний сам не выступает в качестве сервера;
- протоколы — при помощи протоколов клиенты и серверы обмениваются информацией;

- клиентские технологии — Web-приложения по своей природе распределенные, часть работы выполняется на сервере, часть — лишь после того, как страница загружена клиентом.

1.3.1. Серверные технологии

UNIX-подобная операционная система

Все программы, будь то на сервере или на клиенте, работают под управлением операционной системы. В Интернете на серверах в качестве серверной операционной системы используется, как правило, UNIX-подобная операционная система (Linux, FreeBSD, Solaris и т. д.). Большинство пользователей Интернета (клиентов Web-сервера), по крайней мере, в Российской Федерации в качестве операционной системы использует Windows, поэтому при изучении Web-технологий основы UNIX будут крайне полезны, т. к. порядок работы в этих двух операционных системах различается достаточно сильно. Знания UNIX-подобной операционной системы не требуются на начальном этапе обучения, многие обходятся без основ UNIX, уже профессионально занимаясь Web-разработкой. Однако изучение этой операционной системы позволяет более уверенно чувствовать себя в мире Web-разработки, т. к. несмотря на кроссплатформенную независимость языка PHP, различие операционных систем Windows и UNIX дает о себе знать (особенно при работе с файловой системой). Кроме того, для многих параметров сервера не предусматривается Web-интерфейс, и ими можно управлять только через удаленный доступ (SSH), при работе через который необходимы глубокие знания команд операционной системы.

Web-сервер

За формирование Web-страниц на сервере несет ответственность Web-сервер. Существует несколько различных Web-серверов, наиболее известные — IIS-сервер операционной системы Windows и Apache, применяющийся главным образом на UNIX-серверах (впрочем, под управлением Windows Apache также прекрасно работает). При использовании PHP мы будем ориентироваться на Web-сервер Apache, как наиболее распространенный. Сервер Apache, помимо того что реализует протокол HTTP и позволяет обрабатывать обращения клиентов, имеет множество модулей, позволяющих преобразовывать URL-адреса, осуществлять переадресацию, подключать языки программирования (PHP, Perl и т. п.), защищать папки и файлы паролем и многое другое. Сервер Apache и его модули имеют множество директив, которые могут применяться как в основном конфигурационном файле, так и в специальном файле .htaccess, позволяющем настраивать работу файлов в отдельных папках. Изучение принципов работы и директив Web-сервера Apache позволят создавать более эффективные Web-приложения, а также решать множество задач в области управления сайтами, которые можно решить только на уровне Web-сервера.

Серверный язык

Следующим компонентом выступает серверный язык, позволяющий формировать динамические страницы. Практически любой серверный язык можно использовать для создания динамических страниц.

ЗАМЕЧАНИЕ

На заре Интернета, когда язык программирования PHP еще не существовал, авторы данной книги для создания сайтов успешно использовали C++ и даже Fortran.

Однако наибольшее распространение получили специализированные языки, которые либо позволяли легко обрабатывать текст (Perl), либо имели встроенные средства для работы с Web-средой (ASP, PHP, Java). В нашей стране наибольшую популярность приобрел язык программирования PHP, рассматриваемый в данной книге.

Точно так же как и Web-сервер Apache, PHP имеет ядро, где сосредоточены базовые функции, а также множество подключаемых внешних модулей. Наиболее популярные и хорошо зарекомендовавшие себя модули поставляются в официальном дистрибутиве PHP, другие модули можно загрузить в дополнительном архиве или найти на многочисленных сайтах в Интернете. В любом случае язык программирования постоянно перестраивается, некоторые модули устаревают и исключаются из ядра и дистрибутива, другие, получившие популярность, — включаются. Поэтому язык PHP предоставляет собой чрезвычайно гибкий инструмент разработки, подстраивающийся под динамично развивающуюся Web-среду.

PHP — не самый быстрый язык программирования, однако один из самых удобных, что позволяет быстро разрабатывать Web-приложения. В Web-условиях, когда среднее время жизни приложения составляет полгода, а сами приложения постоянно подвергаются модификации, скорость разработки приложений становится решающим фактором. Ранее компьютерное время стоило дороже времени разработчиков (да и программы использовались значительно более долгий срок). В настоящий момент ситуация изменилась кардинально — приобрести быстрый сервер проще и дешевле, чем нанять 10 программистов на C++, разрабатывающих приложение за полгода, вместо одного PHP-программиста, разрабатывающего приложение за один месяц.

ЗАМЕЧАНИЕ

Широкая распространенность PHP вовсе не означает, что все остальные серверные языки следует игнорировать — чем больше языков программирования вы освоите, тем более конкурентоспособными вы будете и тем лучше освоите Web-программирование. Помимо PHP при Web-разработке часто используются Perl, семейство языков ASP.NET (VB, C#, ASP), Java, C/C++, Ruby и Python.

Файлы и базы данных

В любом языке программирования и технологии файлы играют значительную роль, в них хранятся как сами программы, так и данные, которые вводятся пользователями, обрабатываются скриптами и архивируются на сервере. Зачастую к серверу одновременно обращается огромное количество пользователей, что создает определенные трудности при записи информации в один и тот же файл, т. к. сразу несколько потоков стремятся получить доступ к файлам, что при игнорировании этой проблемы может приводить к повреждению записанной в него информации.

Для того чтобы каждый раз при разработке очередного Web-приложения не решать проблемы одновременного доступа к файлу, используются базы данных (или, точнее, системы управления базами данных, СУБД). СУБД разрабатываются десятки лет про-

фессионалами своей области с использованием быстрых компилируемых языков (С, С++). Поэтому СУБД в Web-приложениях всегда быстрее файлового доступа. Кроме того, код с использованием базы данных зачастую в несколько раз меньше по объему (а, следовательно, разрабатывается он в более короткие сроки), чем код, использующий файлы. Чем быстрее вы освоите приемы работы с СУБД, тем более быстродействующим и элегантным будет ваш код, и тем быстрее вы сможете его создавать.

Совместно с PHP может использоваться множество баз данных, начиная коммерческими Oracle и MS SQL, заканчивая свободно-распространяемыми MySQL, PostgreSQL и FireBird. Все СУБД объединяет тот факт, что взаимодействие с ними осуществляется при помощи языка структурированных запросов SQL. Взаимодействие с СУБД сводится к выполнению запросов, построенных при помощи специализированного языка SQL. Язык SQL является общим для всех баз данных — освоив одну из баз данных, вы без труда сможете изучить любую другую.

ЗАМЕЧАНИЕ

Следует все же отметить, что, несмотря на стандартизацию языка SQL, разные базы данных имеют различные SQL-диалекты, иногда достаточно сильно отличающиеся друг от друга.

Одной из самых популярных баз данных, применяемых в Web-разработке, является СУБД MySQL, которая заслужила признательность разработчиков благодаря свободному распространению, высокому быстродействию и надежности, а также богатой функциональности.

Электронная почта

Электронная почта появилась задолго до Web-среды и является в настоящий момент неотъемлемой частью Интернета. Пользователи привыкли отправлять и получать почтовые сообщения. Язык программирования PHP имеет средства как для отправки почтовых сообщений, так и для доступа к почтовым ящикам (локальным или расположенным на удаленном сервере).

1.3.2. Клиентские технологии

Серверные и клиентские технологии разделены в пространстве и времени. PHP выполняется на сервере и формирует страницу, содержащую HTML-разметку, JavaScript-код, Flash-ролики, которые отправляются по протоколу HTTP клиенту. Такая страница интерпретируется браузером и отображается клиенту. В момент просмотра страницы клиентом, PHP, Web-сервер Apache уже отработали и отослали страницу. Заставить их среагировать на какие-то действия пользователя можно, только повторно послав запрос серверу. То же самое касается клиентских технологий, которые не выполняются на сервере — они вступают в действие, только когда страница получена клиентом.

Web-браузеры, HTML

Клиенты Web-серверов используют различные операционные системы и браузеры — программы, предназначенные для просмотра Web-страниц. Наиболее популярные среди них на сегодняшний день — Internet Explorer, FireFox и Opera. Одной из особенностей

является тот факт, что все браузеры (а также их различные версии) по-разному интерпретируют язык разметки HTML, каскадные таблицы стилей CSS и клиентский язык JavaScript. Это требует от Web-разработчиков тестирования сайтов в нескольких браузерах (разных версий), а также зачастую отказа от нововведений, которые поддерживаются одними браузерами и не поддерживаются другими.

Если в области клиентских технологий среди производителей браузеров существуют разногласия, то взаимодействие с серверной частью все браузеры поддерживают очень хорошо. Взаимодействие между браузерами и Web-серверами осуществляется при помощи протокола HTTP.

ЗАМЕЧАНИЕ

Существуют две версии протокола HTTP — 1.0 и 1.1. В настоящий момент везде (за исключением некоторых промежуточных прокси-серверов) используется протокол HTTP версии 1.1.

Каскадные таблицы стилей CSS и XML

Изначально HTML разрабатывался как язык разметки, т. е. язык, описывающий документ независимо от его внешнего вида. Однако очень скоро он стал применяться для создания дизайна, его конструкции стали использоваться для того, чтобы добиться того или иного внешнего эффекта. Под давлением разработчиков Web-страниц и браузеров в HTML вносились все новые и новые элементы. Начались браузерные войны, когда производители браузеров намеренно вносили дополнительные элементы, не совместимые с другими браузерами.

Для того чтобы разделить структуру и оформление документов, консорциумом W3C, координирующим развитие Web, были введен язык разметки XML и каскадные таблицы стилей CSS, при помощи которых можно оформить XML-код. Язык XML настолько гибок, что позволяет самостоятельно определить нужный XML-формат путем разработки собственного словаря. В настоящий момент уже разработано огромное число словарей XML, позволяющих описывать любую информацию — от химических реакций (CML, Chemical Markup Language) до финансовых данных (OFX, Open Financial Exchange). По сравнению с HTML XML является более гибким форматом; повсеместного применения XML еще не получил, но это лишь вопрос времени.

Каскадные таблицы стилей позволяют наложить дизайн на XML-документ. Пока до окончательного разделения структуры и оформления далеко, XML еще не очень популярен в Web в качестве языка разметки. Однако каскадные таблицы стилей применяются очень интенсивно в HTML. Если требуется создание стильных современных дизайнов, без каскадных таблиц стилей CSS не обойтись.

Flash-ролики

Технология Flash позволяет реализовывать на стороне клиента сложные динамические эффекты в виде Flash-роликов. По сути каждый Flash ролик представляет собой мини-сайт, позволяющий демонстрировать сложные графические эффекты, реализовывать игры и т. п. Внешний вид Flash-роликов не зависит от браузера и этим выгодно отличается от связки HTML и CSS. К недостаткам Flash можно отнести значительный размер,