

УДК 681.3.068+800.92РНР 5
ББК 32.973.26-018.1
К73

Котеров, Д. В.

К73 РНР 5 / Д. В. Котеров, А. Ф. Костарев. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 1104 с.: ил. — (В подлиннике)

ISBN 978-5-9775-0315-0

Рассматриваются основы функционирования Web-серверов, сборка исполняемого модуля PHP в ОС UNIX, инструментарий Web-разработчика (в том числе утилиты отладки сценариев), синтаксис и стандартные функции языка. Приведено описание функций PHP для работы с массивами, файлами, СУБД MySQL, регулярными выражениями формата PCRE, графическими примитивами, почтой, сессиями и т. д. Особое внимание уделено новым возможностям языка по работе с XML-документами, объектно-ориентированному программированию, а также подходам к отделению PHP-кода от HTML-шаблонов сайта.

Во втором издании добавлены главы про технологии AJAX и DbSimple, исправлены замеченные опечатки.

Для Web-программистов

УДК 681.3.068+800.92РНР 5
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>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</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 21.07.08.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 89,01.

Тираж 2500 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.003650.04.08 от 14.04.2008 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов

в ОАО "Техническая книга"

190005, Санкт-Петербург, Измайловский пр., 29

ISBN 978-5-9775-0315-0

© Котеров Д. В., Костарев А. Ф., 2008

© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

Предисловие	1
Для кого написана эта книга	2
Сайт книги.....	3
Исправления во втором издании	3
Общая структура книги	3
Часть I	4
Часть II	4
Часть III	5
Часть IV	5
Часть V	5
Часть VI	6
Часть VII	6
Листинги	7
Предметный указатель	8
Как создавалась книга	8
Благодарности	10
 Предисловие к первому изданию.....	 11
 ЧАСТЬ I. ОСНОВЫ WEB-ПРОГРАММИРОВАНИЯ.....	 13
 Глава 1. Принципы работы Интернета.....	 15
Протоколы передачи данных	15
Семейство TCP/IP.....	17
Адресация в Сети	18
IP-адрес	18
Доменное имя	19
Порт	21
Установка соединения	22
Обмен данными	23
Терминология	23
Сервер.....	23
Узел.....	24
Порт.....	24
Сетевой демон, сервис, служба	24
Хост	25
Виртуальный хост	25
Провайдер	26
Хостинг-провайдер (хостер)	26
Хостинг.....	26
Сайт	26
HTML-документ	27
Страница (или HTML-страница)	27
Скрипт, сценарий	27
Web-программирование	27

Взаимосвязь терминов	28
World Wide Web и URL	28
Протокол	29
Имя хоста	29
Порт	29
Путь к странице	29
Резюме	30
Глава 2. Интерфейс CGI и HTTP	31
Что такое CGI?	31
Секреты URL	32
Заголовок запроса и метод <i>GET</i>	33
<i>GET</i>	34
<i>POST</i>	35
<i>Content-Type</i>	35
<i>Host</i>	35
<i>User-Agent</i>	36
<i>Referer</i>	36
<i>Content-length</i>	37
<i>Cookie</i>	37
<i>Accept</i>	37
Эмуляция браузера через telnet	37
Метод <i>POST</i>	38
URL-кодирование	39
Проблема русских кодировок	39
Что такое кодировка символов?	39
"Русский Apache" и кодировка	41
Пример сбойной конфигурации	41
Отключение автоматической перекодировки	42
Что такое формы и для чего они нужны?	42
Передача параметров "вручную"	43
Использование формы	43
Абсолютный и относительный пути к сценарию	45
Метод <i>POST</i> и формы	45
Кодировка входных данных	46
Резюме	47
Глава 3. CGI изнутри	48
Язык C	48
Работа с исходными текстами на C	49
Компиляция программ	49
Передача документа пользователю	50
Заголовки ответа	50
Заголовков кода ответа	50
"Подделывание" заголовка ответа	51
<i>Content-type</i>	51
<i>Pragma</i>	51
<i>Location</i>	52
<i>Set-cookie</i>	52
<i>Date</i>	52
<i>Server</i>	52
Примеры CGI-сценариев на C	53
Вывод бинарного файла	54

Передача информации CGI-сценарию.....	54
Переменные окружения.....	54
Передача параметров методом <i>GET</i>	56
Передача параметров методом <i>POST</i>	57
Расшифровка URL-кодированных данных.....	58
Формы.....	61
Тег <i><input></i> — различные поля ввода.....	62
Текстовое поле (<i>text</i>).....	62
Поле ввода пароля (<i>password</i>).....	63
Скрытое текстовое поле (<i>hidden</i>).....	63
Независимый переключатель (<i>checkbox</i>).....	64
Зависимый переключатель (<i>radio</i>).....	65
Кнопка отправки формы (<i>submit</i>).....	65
Кнопка сброса формы (<i>reset</i>).....	66
Рисунок для отправки формы (<i>image</i>).....	66
Тег <i><textarea></i> — многострочное поле ввода текста.....	66
Тег <i><select></i> — список.....	67
Списки множественного выбора (<i>multiple</i>).....	67
HTML-сущности.....	68
Загрузка файлов.....	69
Формат данных.....	69
Тег загрузки файла (<i>file</i>).....	71
Что такое cookies и "с чем их едят"?.....	72
Установка cookie.....	74
Получение cookies из браузера.....	75
Пример программы для работы с cookies.....	76
Авторизация.....	77
Резюме.....	78

ЧАСТЬ II. ВЫБОР И НАСТРОЙКА ИНСТРУМЕНТАРИЯ..... 79

Глава 4. Установка Apache..... 81

Традиционный процесс отладки сайта.....	81
Локальный сервер.....	82
Почему Apache?.....	82
От слов к делу: установка Apache.....	83
Получение дистрибутива.....	83
Получение документации.....	83
Создание виртуального диска.....	84
Установка Apache.....	85
Настройка файла конфигурации Apache.....	86
Запуск и остановка.....	89
Тестирование и устранение неполадок.....	90
Дополнительные драйверы для Windows 95.....	90
Устранение неполадок.....	90
Проверка HTML-страниц.....	91
Проверка SSI.....	91
Виртуальные хосты Apache.....	92
Разновидности виртуальных хостов.....	92
Именованые виртуальных хостов.....	92
Параметры хостов.....	92
Что получилось в итоге?.....	96

Apache для Windows и безопасность.....	96
Службы.....	97
Безопасность.....	97
Пример уязвимого скрипта.....	98
Заключительные слова о безопасности.....	99
Ссылки.....	100
Резюме.....	100
Глава 5. Установка PHP и MySQL.....	101
Установка PHP.....	101
Дополнительные драйверы для Windows 98.....	102
Получение документации.....	102
Состав дистрибутива PHP.....	103
CGI-версия PHP.....	103
Консольная версия PHP.....	103
"Бесконсольная" версия.....	104
Динамическая библиотека Apache.....	104
Файл конфигурации php.ini-dist.....	104
Библиотеки расширения.....	104
Другие файлы.....	105
Конфигурирование PHP.....	105
Конфигурирование расширений.....	106
Настройка Apache для работы с PHP.....	107
Установка PHP в виде CGI-программы.....	108
Установка PHP в виде модуля сервера.....	108
Тестирование PHP.....	111
Проверка конфигурации.....	112
Установка MySQL.....	112
Получение дистрибутива.....	112
Получение документации.....	113
Конфигурирование MySQL.....	113
Настройка параметров сервера.....	113
Запуск и остановка.....	115
Тестирование MySQL.....	116
Отладка и устранение ошибок.....	117
Отключение межсетевого экрана (firewall).....	117
Просмотр истории обращения к файлам.....	117
Просмотр заголовков HTTP.....	118
Работа с интерактивным отладчиком PHPed.....	120
Традиционная отладка PHP-сценариев.....	120
Интерактивный отладчик.....	121
Отладчик PHPed.....	122
Настройка PHPed.....	122
Ядро отладчика.....	123
"Слушатель" отладчика.....	123
Интерактивная оболочка отладчика.....	125
Тестирование PHPed.....	127
Ссылки.....	128
Резюме.....	129
Глава 6. Денвер: автоматизация установки инструментария.....	130
Что такое Денвер?.....	130
Условия использования.....	131

Что входит в Денвер?	131
Состав базового пакета	131
Дополнительные пакеты расширения	131
Поддержка разработчиков	132
Установка дистрибутива	132
Подготовка к работе с сетью	132
Пользователю Windows 95	133
Инсталляция	134
Работа с виртуальными хостами	136
Проблемы с контроллером удаленного доступа	138
Проблемы с прокси-сервером	139
Вопросы и ответы	139
Ссылки	140
Резюме	140

Глава 7. Установка PHP 5 в ОС Unix.....141

PHP 5 как CGI-приложение	141
Загрузка дистрибутивов	142
Работа с Midnight Commander.....	142
Компиляция программ	143
Распаковка дистрибутивов.....	143
Компиляция библиотеки libxml2	144
Компиляция библиотеки libxslt	145
Компиляция PHP 5	145
Компиляция дополнительных модулей	146
Подключение PHP на машине хостинг-провайдера	146
Сборка PHP в виде модуля Apache.....	147
Сводка команд Unix	147
Ссылки	149
Резюме	149

ЧАСТЬ III. ОСНОВЫ ЯЗЫКА PHP.....151

Глава 8. Характеристика языка PHP.....153

Интерпретатор или компилятор?.....	154
Основные термины	154
Как работает PHP.....	155
Достоинства и недостатки интерпретатора	156
PHP и СУБД	157
Интерпретатор и компилятор.....	158
PHP версии 5	158
Пример PHP-программы.....	159
Использование PHP в Web.....	163
Резюме	164

Глава 9. Переменные, константы, типы данных.....165

Переменные	165
Копирование переменных	166
Типы переменных	166
<i>integer</i> (целое число)	166
<i>double</i> (вещественное число).....	166
<i>string</i> (строка текста).....	167
<i>array</i> (ассоциативный массив).....	167
<i>object</i> (ссылка на объект).....	167

<i>resource</i> (ресурс)	167
<i>boolean</i> (логический тип).....	168
<i>NULL</i> (специальное значение)	168
Действия с переменными	168
Присвоение значения.....	168
Проверка существования	169
Уничтожение.....	169
Определение типа переменной	170
Установка типа переменной.....	171
Оператор присваивания.....	171
Ссылочные переменные	172
Жесткие ссылки.....	172
"Сбор мусора"	173
Символические ссылки.....	173
Ссылки на объекты	174
Некоторые условные обозначения	175
Константы.....	177
Предопределенные константы	177
Определение констант	178
Проверка существования константы	178
Отладочные функции.....	179
Резюме.....	181
Глава 10. Выражения и операции РНР	182
Выражения.....	182
Логические выражения	183
Строковые выражения	184
Строка в апострофах	184
Строка в кавычках.....	184
Неге-документ.....	186
Вызов внешней программы	186
Операции	187
Арифметические операции	187
Строковые операции	187
Операции присваивания	188
Операции инкремента и декремента	188
Битовые операции	189
Операции сравнения	189
Особенности операторов <code>==</code> и <code>!=</code>	189
Сравнение сложных переменных.....	190
Операция эквивалентности	191
Логические операции	193
Операция отключения предупреждений	193
Особенности оператора <code>@</code>	195
Противопоказания к использованию	195
Резюме.....	196
Глава 11. Работа с данными формы.....	197
Передача данных командной строки.....	197
Формы	199
Трансляция полей формы.....	200
Трансляция переменных окружения	202
Трансляция cookies.....	202
Обработка списков.....	203
Обработка массивов	204

Диагностика.....	205
Режим <i>register_globals</i>	205
Первый пример уязвимости	206
Второй пример уязвимости	206
Порядок трансляции переменных	207
Особенности флажков <i>checkbox</i>	208
Резюме.....	209
Глава 12. Конструкции языка	210
Инструкция <i>if-else</i>	210
Использование альтернативного синтаксиса.....	211
Цикл с предусловием <i>while</i>	212
Цикл с постусловием <i>do-while</i>	212
Универсальный цикл <i>for</i>	213
Инструкции <i>break</i> и <i>continue</i>	214
Нетрадиционное использование <i>do-while</i> и <i>break</i>	215
Цикл <i>foreach</i>	216
Конструкция <i>switch-case</i>	217
Инструкции <i>require</i> и <i>include</i>	218
Инструкции однократного включения.....	219
Суть проблемы.....	219
Решение: <i>require_once</i>	221
Другие инструкции	221
Резюме.....	222
Глава 13. Ассоциативные массивы	223
Создание массива "на лету". Автомассивы.....	224
Оператор <i>list()</i>	225
Списки и ассоциативные массивы: путаница?.....	226
Оператор <i>array()</i> и многомерные массивы.....	226
Операции над массивами	227
Доступ по ключу.....	228
Функция <i>count()</i>	228
Слияние массивов	228
Слияние списков.....	229
Обновление элементов.....	229
Косвенный перебор элементов массива	230
Перебор списка	230
Перебор ассоциативного массива	231
Недостатки косвенного перебора.....	231
Прямой перебор массива.....	232
Старый способ перебора.....	232
Перебор в стиле PHP 4.....	233
Ссылочный синтаксис <i>foreach</i>	233
Списки и строки	234
Сериализация	235
Упаковка.....	236
Распаковка	236
Резюме.....	236
Глава 14. Функции и области видимости.....	237
Пример функции.....	238
Общий синтаксис определения функции	239
Инструкция <i>return</i>	239

Объявление и вызов функции	240
Параметры по умолчанию	241
Передача параметров по ссылке	241
Переменное число параметров	243
Локальные переменные	245
Глобальные переменные	245
Массив <i>\$GLOBALS</i>	246
Самовложенность	247
Как работает инструкция <i>global</i>	247
Статические переменные	248
Рекурсия	249
Факториал	249
Пример функции: <i>dumper()</i>	250
Вложенные функции	251
Условно определяемые функции	252
Эмуляция функции <i>virtual()</i>	253
Передача функций "по ссылке"	254
Использование <i>call_user_func()</i>	255
Использование <i>call_user_func_array()</i>	255
Возврат функцией ссылки	256
Технология отложенного копирования	257
Несколько советов по использованию функций	259
Резюме	260

ЧАСТЬ IV. СТАНДАРТНЫЕ ФУНКЦИИ PHP.....261

Глава 15. Строковые функции263

Конкатенация строк	263
О сравнении строк	264
Особенности <i>strcmp()</i>	265
Работа с одиночными символами	266
Отрезание пробелов	267
Базовые функции	268
Работа с подстроками	269
Замена	270
Подстановка	271
Преобразования символов	273
Изменение регистра	275
Установка локали (локальных настроек)	276
Преобразование кодировок	277
Функции форматных преобразований	278
Форматирование текста	280
Работа с бинарными данными	281
Хэш-функции	283
Сброс буфера вывода	285
Резюме	285

Глава 16. Работа с массивами286

Лексикографическая и числовая сортировки	286
Сортировка произвольных массивов	287
Сортировка по значениям	287
Сортировка по ключам	287
Пользовательская сортировка по ключам	288

Пользовательская сортировка по значениям	289
Перевоорачивание массива	289
"Естественная" сортировка	290
Сортировка списков	291
Сортировка списка	291
Пользовательская сортировка списка	292
Перемешивание списка	292
Ключи и значения	293
Слияние массивов	294
Работа с подмассивами	294
Работа со стеком и очередью	295
Переменные и массивы	296
Применение в шаблонах	298
Создание диапазона чисел	299
Работа с множествами	299
Пересечение	299
Разность	300
Объединение	300
Резюме	301
Глава 17. Математические функции	302
Встроенные константы	302
Функции округления	303
Случайные числа	303
Перевод в различные системы счисления	306
Минимум и максимум	307
Не-числа	307
Степенные функции	308
Тригонометрия	308
Резюме	309
Глава 18. Работа с файлами	310
О текстовых и бинарных файлах	310
Открытие файла	311
Конструкция <i>or die()</i>	313
Различия текстового и бинарного режимов	313
Сетевые соединения	314
Прямые и обратные слэши	314
Безымянные временные файлы	315
Закрытие файла	316
Чтение и запись	316
Блочные чтение/запись	317
Построчные чтение/запись	317
Чтение CSV-файла	318
Проблемы стандартной функции <i>fgetcsv()</i>	319
Положение указателя текущей позиции	319
Работа с путями	320
Манипулирование целыми файлами	322
Чтение и запись целого файла	323
Чтение INI-файла	324
Другие функции	325
Блокирование файла	326
Рекомендательная и жесткая блокировки	326
Функция <i>flock()</i>	327

Типы блокировок	327
Исключительная блокировка	327
Разделяемая блокировка	331
Блокировки с запретом "подвисания"	332
Пример счетчика	333
Резюме	334
Глава 19. Права доступа и атрибуты файлов	335
Идентификатор пользователя	335
Идентификатор группы	336
Владелец файла	337
Права доступа	337
Числовое представление прав доступа	338
Особенности каталогов	339
Примеры	340
Домашний каталог пользователя	340
Защищенный от записи файл	340
CGI-скрипт	341
Системные утилиты	341
Закрытые системные файлы	341
Права доступа на RHP-сценарии	341
Функции RHP	342
Права доступа	342
Определение атрибутов файла	343
Специальные функции	344
Определение типа файла	345
Определение возможности доступа	345
Ссылки	346
Символические ссылки	346
Жесткие ссылки	347
Резюме	348
Глава 20. Работа с каталогами	349
Манипулирование каталогами	349
Работа с записями	350
Пример: печать дерева каталогов	351
Получение содержимого каталога	352
Резюме	353
Глава 21. Запуск внешних программ	354
Запуск утилит	354
Оператор "обратные апострофы"	356
Экранирование командной строки	356
Режим <i>safe_mode</i>	357
Каналы	358
Временные файлы	359
Открытие канала	359
Взаимная блокировка (deadlock)	360
Резюме	361
Глава 22. Работа с датами и временем	362
Представление времени в формате timestamp	362
Вычисление времени работы скрипта	363
Большие вещественные числа	363

Построение строкового представления даты	364
Построение timestamp	366
Разбор timestamp	368
Григорианский календарь	368
Проверка даты	369
Календарик	370
Дата и время по Гринвичу	372
Время по GMT	372
Хранение абсолютного времени	372
Перевод времени	373
Окончательное решение задачи	374
Резюме	375

Глава 23. Управление интерпретатором.....376

Информационные функции	376
Настройка параметров PHP	377
PHP в виде модуля Apache	378
CGI-версия PHP	378
Создание переадресации для интерпретатора PHP	379
Переадресация в Unix	380
Использование функции <i>ini_set()</i>	381
Некоторые популярные директивы	381
Контроль ошибок	386
Директивы контроля ошибок	386
Установка режима вывода ошибок	387
Оператор отключения ошибок	388
Пример использования оператора @	389
Предостережения	389
Перехват ошибок	390
Проблемы с оператором @	392
Генерация ошибок	393
Стек вызовов функций	393
Принудительное завершение программы	394
Финализаторы	395
Генерация кода во время выполнения	396
Выполнение кода	396
Генерация функций	398
Другие функции	400
Резюме	400

Глава 24. Основы регулярных выражений в формате PCRE401

Начнем с примеров	401
Пример первый	401
Пример второй	402
Пример третий	402
Пример четвертый	403
What is the PCRE?	404
Терминология	404
Языки регулярных выражений	405
Использование регулярных выражений в PHP	406
Сопоставление	406
Сопоставление с заменой	407

Язык PCRE	408
Ограничители	408
Альтернативные ограничители	409
Отмена действия спецсимволов	409
Простые символы (литералы)	410
Классы символов	411
Альтернативы	411
Отрицательные классы	412
Квантификаторы повторений	413
Ноль или более совпадений	413
Одно или более совпадений	413
Ноль или одно совпадение	414
Заданное число совпадений	414
Мнимые символы	414
Оператор альтернативы	415
Группирующие скобки	415
"Карманы"	416
Использование карманов в функции замены	417
Использование карманов в функции сопоставления	418
"Жадность" квантификаторов	419
Рекуррентные структуры	420
Группировка без захвата	421
Модификаторы	421
Модификатор /i: игнорирование регистра	421
Модификатор /x: пропуск пробелов и комментариев	422
Модификатор /m: многострочность	422
Модификатор /s: (одноточный поиск)	423
Модификатор /e: выполнение PHP-программы при замене	423
Незахватывающий поиск	424
Позитивный просмотр вперед	424
Негативный просмотр вперед	425
Позитивный просмотр назад	425
Негативный просмотр назад	426
Другие возможности PCRE	426
Функции PHP	426
Поиск совпадений	426
Замена совпадений	429
Разбиение по регулярному выражению	431
Выделение всех уникальных слов из текста	431
Экранирование символов	432
Фильтрация массива	433
Примеры использования регулярных выражений	434
Преобразование адресов e-mail	434
Преобразование гиперссылок	435
Быть или не быть?	436
Ссылки	436
Резюме	436
Глава 25. Работа с HTTP и WWW	437
Заголовки ответа	437
Вывод заголовка ответа	437
Проблемы с заголовками	437
Запрет кэширования	438
Получение выведенных заголовков	439

Получение заголовков запроса.....	440
Работа с cookies.....	440
Немного теории.....	440
Установка cookie.....	441
Массивы и cookie.....	442
Получение cookie.....	443
SSI и функция <i>virtual()</i>	443
Эмуляция функции <i>virtual()</i>	444
Разбор URL.....	445
Разбиение и "склеивание" <i>QUERY_STRING</i>	445
Разбиение и "склеивание" URL.....	446
Пример.....	448
Резюме.....	449
Глава 26. Сетевые функции.....	450
Сеть и файловые функции.....	450
Проблемы безопасности.....	451
Другие схемы.....	451
Работа с сокетами.....	451
"Эмуляция" браузера.....	452
Неблокирующее чтение.....	453
Функции для работы с DNS.....	453
Преобразование IP-адреса в доменное имя и наоборот.....	454
Получение MX-записи.....	455
Резюме.....	457
Глава 27. Посылка писем через PHP.....	458
Формат электронного письма.....	458
Отправка письма.....	459
Почтовые шаблоны.....	460
Расщепление заголовков.....	461
Анализ заголовков.....	462
Русскоязычные кодировки.....	463
Заголовок <i>Content-type</i> и кодировка.....	464
Кодировка заголовков.....	464
Кодирование тела письма.....	466
Письма с вложениями.....	467
Динамическая смена кодировки.....	467
Активные шаблоны.....	468
Настройки PHP.....	470
Ссылки.....	471
Резюме.....	471
Глава 28. Работа с СУБД MySQL.....	472
Что такое база данных?.....	472
Неудобство работы с файлами.....	473
Архитектура MySQL.....	474
Администрирование базы данных.....	475
Порядок работы с базой данных.....	475
Интерфейсы для работы с MySQL.....	476
Соединение с сервером.....	476
Обработка ошибок.....	477
Выполнение запросов к базе данных.....	477

Автоматизация подключения к СУБД	478
Создание нового пользователя	480
Подключение с правами администратора	480
Язык запросов СУБД MySQL	480
<i>CREATE DATABASE</i> : создание базы данных	481
<i>CREATE TABLE</i> : создание таблицы	481
Типы полей	482
Модификаторы и флаги типов	485
<i>DROP TABLE</i> : удаление таблицы	485
<i>INSERT</i> : вставка записи в таблицу	486
<i>DELETE</i> : удаление записей	486
<i>SELECT</i> : поиск и выборка записей	486
Получение числа записей, удовлетворяющих выражению	487
Получение уникальных значений столбцов	487
<i>UPDATE</i> : обновление записей	488
Комментарии	488
Получение результата	488
Преобразование result-set в двумерный массив	489
<i>AS</i> : переименование полей	490
Передвижение по результирующему набору	490
<i>LIMIT</i> : ограничение выборки средствами SQL	491
Выборка строки в виде списка	491
Параметры результата	491
Получение отдельной ячейки результата	492
Информация о результате	492
Пример использования функций	493
Информация о таблицах и полях	494
<i>AUTO_INCREMENT</i> : уникальные идентификаторы	495
Плотное следование идентификаторов	496
Получение идентификатора до вставки	497
Индексы и производительность БД	497
Как "работают" индексы	498
Создание индексов	499
<i>EXPLAIN</i> : план выполнения запроса	499
Недостатки индексов	499
MySQL и проблемы безопасности	500
Суть проблемы	500
Экранирование спецсимволов	500
Шаблоны запросов и placeholders	501
Пример: гостевая книга	504
Ссылки	507
Резюме	508
Глава 29. Управление сессиями	509
Что такое сессия?	510
Зачем нужны сессии?	510
Механизм работы сессий	511
Инициализация сессии	512
Пример использования сессии	512
Уничтожение сессии	514
Сессии и cookies	514
Явное использование константы <i>SID</i>	514
Неявное изменение гиперссылок	515
Неявное изменение формы	516

Использовать ли cookies в сессиях?	517
"Лишние" идентификаторы	518
Идентификатор сессии и имя группы	518
Имя группы сессий	518
Идентификатор сессии	519
Путь к временному каталогу	520
Стоит ли изменять группу сессий?	520
Установка обработчиков сессии	521
Обзор обработчиков	521
Регистрация обработчиков	522
Пример: переопределение обработчиков	523
Регистрация глобальных переменных	525
Резюме	526
Глава 30. Работа с изображениями	527
Библиотека GD и формат GIF	528
Универсальная функция <i>getimagesize()</i>	528
Работа с изображениями и библиотека GD	530
Пример создания изображения	530
Создание изображения	531
Загрузка изображения	532
Определение параметров изображения	532
Сохранение изображения	533
Преобразование изображения в палитровое	534
Работа с цветом в формате RGB	534
Создание нового цвета	535
Текстовое представление цвета	535
Получение ближайшего в палитре цвета	535
Эффект прозрачности	536
Получение RGB-составляющих	537
Использование полупрозрачных цветов	537
Графические примитивы	538
Копирование изображений	538
Прямоугольники	540
Выбор пера	540
Линии	541
Дуга сектора	541
Закраска произвольной области	542
Закраска текстурой	542
Многоугольники	542
Работа с пикселями	543
Работа с фиксированными шрифтами	544
Загрузка шрифта	544
Параметры шрифта	545
Вывод строки	545
Работа со шрифтами TrueType	545
Вывод строки	546
Проблемы с русскими буквами	546
Определение границ строки	547
Коррекция функции <i>imageTtfBBox()</i>	547
Пример	549
Ссылки	551
Резюме	551

ЧАСТЬ V. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА PHP.....553

Глава 31. Организация библиотек.....555

Подключение файла библиотеки	556
Корневой каталог библиотек	556
Несколько путей поиска	557
Файл конфигурации	558
Преимущества использования путей подключения	558
Разрешение конфликтов имен	559
Проблема именования функций	559
Пространства имен	560
Соглашения PEAR	561
Соглашение 1: классы вместо пространств имен	561
Функции библиотеки	562
Переменные в библиотеке	562
Константы в библиотеке	562
Пространство имен <i>self</i>	563
Соглашение 2: формат библиотеки	563
Соглашение 3: использование подкаталогов	565
Автоматическая загрузка классов	566
Библиотека поддержки автозагрузки	566
Каталог модуля	569
PEAR: преобразование имени класса в имя файла	569
Неприятная особенность <i>require_once</i>	573
Собираем все вместе	574
Пример использования всех библиотек	575
Главный файл скрипта	576
Недостатки глобальных переменных	576
Использование функций	577
Интерфейс библиотеки	578
Наследование и расширение модулей	579
Совместимость PHP 5 и PHP 4	581
Резюме	581

Глава 32. Классы и сокрытие данных582

Новые возможности PHP 5	582
Класс как тип данных	583
Создание нового класса	584
Отличие классов от библиотек	585
Работа с классами	586
Создание объекта некоторого класса	586
Доступ к свойствам объекта	586
Доступ к методам	587
Создание нескольких объектов	588
Перегрузка преобразования в строку	589
Инициализация и разрушение	590
Конструктор	590
Параметры по умолчанию	591
Старый способ создания конструктора	592
Деструктор	592
Вопрос освобождения ресурсов	592
Описание деструктора	594

Алгоритм сбора мусора	596
Циклические ссылки	597
Проблема циклических ссылок	599
Решение проблемы циклических ссылок	600
Принудительное удаление объектов	600
Права доступа к членам класса	600
Модификаторы доступа	601
<i>Public</i> : открытый доступ	601
<i>Private</i> : доступ только из методов класса	601
<i>Protected</i> : доступ из методов производного класса	602
Неявное объявление свойств	602
Статические члены класса	603
Пример: счетчик объектов	603
Пример: кэш ресурсов	604
Общие рекомендации	605
Перехват обращений к членам класса	606
Клонирование объектов	608
Переопределение операции клонирования	609
Запрет клонирования	610
Перехват сериализации	610
Резюме	611
Глава 33. Наследование и виртуальные методы	612
Расширение класса	613
Метод включения	613
Недостатки метода	614
Несовместимость типов	615
Наследование	616
Переопределение методов	617
Модификаторы доступа при переопределении	617
Доступ к методом базового класса	617
Финальные методы	618
Запрет наследования	619
Константы <code>__CLASS__</code> и <code>__METHOD__</code>	619
Полиморфизм	619
Абстрагирование	620
Виртуальные методы	622
Расширение иерархии	625
Абстрактные классы и методы	625
Совместимость родственных типов	627
Уточнение типа в функциях	627
Оператор <code>instanceof</code>	628
Обратное преобразование типа	628
Множественное наследование и интерфейсы	629
Интерфейсы	629
Множественная реализация интерфейсов	630
Интерфейсы и абстрактные классы	631
Резюме	631
Глава 34. Обработка ошибок и исключения	633
Что такое ошибка?	633
Роли ошибок	634
Виды ошибок	634

Несерьезные ошибки	635
Серьезные ошибки	636
Прекращение выполнения программы.....	636
Возврат недопустимого значения.....	636
Ненормальное состояние программы.....	637
Вызов функции-обработчика	637
Исключения.....	638
Базовый синтаксис	638
Инструкция <i>throw</i>	639
Раскрутка стека.....	640
Исключения и деструкторы.....	641
Исключения и <i>set_error_handler()</i>	642
Классификация и наследование.....	643
Базовый класс <i>Exception</i>	644
Использование интерфейсов	646
Блоки-финализаторы	648
Неподдерживаемая конструкция <i>try...finally</i>	648
"Выделение ресурса есть инициализация"	649
Перехват всех исключений	649
Трансформация ошибок	651
Серьезность "несерьезных" ошибок	651
Преобразование ошибок в исключения	653
Пример	653
Код библиотеки <i>PHP_Exceptionizer</i>	654
Иерархия исключений	657
Фильтрация по типам ошибок	658
Перспективы	658
Резюме.....	659
Глава 35. Отражения, итераторы, массивы	660
Неявный доступ к классам и методам	660
Неявный вызов метода.....	660
Неявный список аргументов	661
Инстанцирование классов.....	662
Использование неявных аргументов.....	662
Аппарат отражений	663
Функция: <i>ReflectionFunction</i>	664
Параметр функции: <i>ReflectionParameter</i>	666
Класс: <i>ReflectionClass</i>	667
Наследование и отражения.....	669
Свойство класса: <i>ReflectionProperty</i>	671
Метод класса: <i>ReflectionMethod</i>	671
Библиотека расширения: <i>ReflectionExtension</i>	672
Различные утилиты: <i>Reflection</i>	673
Исключение: <i>ReflectionException</i>	673
Иерархия	673
Итераторы	673
Стандартное поведение <i>foreach</i>	674
Определение собственного итератора.....	674
Как PHP обрабатывает итераторы	677
Множественные итераторы	678
Виртуальные массивы.....	678
Библиотека SPL.....	681
Резюме.....	681

ЧАСТЬ VI. XML В PHP 5683**Глава 36. Фундамент XML.....685**

XML-расширения языка PHP.....	686
Основные понятия XML.....	688
Типы XML-документов.....	695
Язык XHTML.....	697
Резюме.....	697

Глава 37. DOM1 — объектная модель XML-документа698

Перечень стандартов DOM.....	698
Кодировки.....	700
Класс <i>domDocument</i> , загрузка и выгрузка XML-документов.....	704
Обобщенный класс <i>domNode</i>	708
Классы <i>NodeList</i> и <i>NamedNodeMap</i>	713
Класс <i>NodeList</i>	713
Класс <i>NamedNodeMap</i>	716
Пример программы отображения свойств узлов XML-документа.....	718
Свойства объектов подклассов класса <i>domNode</i>	728
Свойства класса <i>domDocument</i>	730
Свойства класса <i>domDocumentType</i>	734
Свойства класса <i>domEntity</i>	736
Свойства класса <i>domElement</i>	738
Свойства класса <i>domDocumentFragment</i>	740
Свойства класса <i>domAttr</i>	741
Свойства класса <i>domProcessingInstruction</i>	743
Свойства класса <i>domCharacterData</i>	745
Свойства класса <i>domText</i>	746
Свойства класса <i>domCDATASection</i>	747
Свойства класса <i>domComment</i>	748
Построение и корректировка XML-документа.....	749
Создание экземпляра класса стандарта DOM.....	750
Методы класса <i>domElement</i> для работы с атрибутами.....	754
Методы класса <i>domCharacterData</i> для работы с текстом.....	756
Корректировка дерева узлов.....	760
Выборка узлов из документа.....	767
"Живые" объекты.....	769
Резюме.....	774

Глава 38. DOM2 — пространства имен775

Пространство имен.....	775
Не очень простой пример.....	776
Пересечения имен.....	784
Использование пространства имен в атрибутах.....	789
Дополнительные свойства стандарта DOM2.....	790
Дополнительные методы стандарта DOM2.....	791
Методы создания элементов и атрибутов в указанном пространстве имен.....	791
Методы работы с атрибутами элемента класса <i>domElement</i>	797
Методы выбора элементов по имени.....	797
Методы доступа к текущим соотношениям префиксов и областей имен.....	798
Дополнительные методы класса <i>NamedNodeMap</i>	802
Дополнительные методы других классов.....	802
Интерфейс <i>domImplementation</i>	802
Перенос узлов между документами.....	804

Нормализация узлов документа	806
Обработка исключительных ситуаций в DOM	806
Резюме	811

Глава 39. DOM3 и другие стандарты.....812

Модули стандарта DOM	812
Дополнительные интерфейсы и методы стандарта DOM3	816
Свойства и методы класса <i>domDocument</i>	817
Свойства класса <i>domNode</i>	818
Свойства класса <i>Entity</i>	818
Дополнительные интерфейсы	818
Методы обработки HTML-документов класса <i>domDocument</i>	818
Методы <i>loadHTMLFile()</i> , <i>loadHTML()</i>	819
Методы <i>saveHTML()</i> , <i>saveHTMLFile()</i>	824
Поддержка различных кодировок в методах обработки HTML-документов	825
Резюме	827

Глава 40. Пути-дорожки: язык XPath828

Базовые понятия языка XPath	828
Программа отображения результата XPath-запроса	830
Структура запроса XPath	837
Оси 838	
Ось <i>child</i>	839
Ось <i>descendant</i>	843
Ось <i>descendant-or-self</i>	845
Ось <i>self</i>	847
Ось <i>following-sibling</i>	848
Ось <i>following</i>	848
Ось <i>attribute</i>	850
Ось <i>parent</i>	852
Ось <i>ancestor</i>	854
Ось <i>ancestor-or-self</i>	854
Ось <i>preceding-sibling</i>	854
Ось <i>preceding</i>	857
Функции	857
Функции узлов	857
Строковые функции	859
Арифметические функции	862
Поддержка областей имен в языке XPath	863
Поддержка областей имен в функциях узлов	865
Привязка префиксов запроса	866
Резюме	871

Глава 41. Расширение SIMPLEXML.....872

Простой пример	872
Загрузка и сохранение XML-документов	874
Доступ к узлам документа	875
Доступ к атрибутам узла	876
Доступ к атрибутам узла по имени	876
Доступ к атрибутам узла в порядке следования	876
Доступ к дочерним узлам	878
Доступ к дочерним узлам по имени	878
Доступ к дочерним узлам в порядке следования	880
Доступ к элементам по выражению языка XPath	881

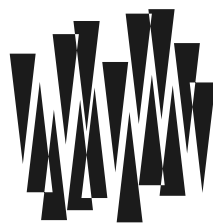
Корректировка документа	882
Резюме	883
Глава 42. Расширение XSLT	884
Общие сведения о языке XSLT	884
Пример XSLT-трансформации	888
Программа songs.xsl (нулевой уровень)	890
Программа song.xsl (первый уровень)	894
Программы второго уровня	897
Программы третьего уровня	901
Объекты и методы поддержки XSLT в PHP 5	902
Метод <i>transformtoXML()</i>	903
Метод <i>transformtoURI()</i>	907
Метод <i>transformtoDOC()</i>	908
Форматы выходных документов	912
Вызов PHP-функций из PHP-программ	916
Поддержка расширений XSLT (EXSLT)	920
Модуль <i>Common</i>	923
Модуль <i>Math</i>	928
Модуль <i>Sets</i>	930
Модуль <i>Functions</i>	933
Модуль <i>Dates and Times</i>	936
Модуль <i>Strings</i>	941
Модуль <i>Regular Expressions</i>	944
Модуль <i>Random</i>	944
Ссылки	945
Резюме	945
ЧАСТЬ VII. ПРИЕМЫ ПРОГРАММИРОВАНИЯ НА PHP 5	947
Глава 43. Загрузка файлов на сервер	949
<i>Multipart</i> -формы	950
Тег выбора файла	950
Закачка файлов и безопасность	950
Поддержка закачки в PHP	951
Простые имена полей закачки	951
Получение закачанного файла	953
Пример: фотоальбом	954
Сложные имена полей	955
Резюме	957
Глава 44. Использование перенаправлений	958
Внешний редирект	958
Внутренний редирект	959
Самопереадресация	961
Резюме	964
Глава 45. Перехват выходного потока	965
Функции перехвата	965
Стек буферов	966
Недостатки "ручного" перехвата	967
Использование объектов и деструкторов	968

Класс для перехвата выходного потока	969
Недостатки класса	971
Проблемы с отладкой	972
Обработчики буферов	972
GZip-сжатие	973
Печать эффективности сжатия	974
Резюме	976
Глава 46. Код и шаблон страницы	977
Первый способ: "вкрапление" HTML в код	977
Второй способ: вставка кода в шаблон	979
Третий способ: Model—View—Controller	980
Шаблон (View)	981
Контроллер (Controller)	981
Модель (Model)	983
Взаимодействие элементов	984
Активные и пассивные шаблоны	985
Активные шаблоны	986
Пассивные шаблоны	986
Недостатки MVC	989
Четвертый способ: компонентный подход	990
Блочная структура Web-страниц	991
Взаимодействие элементов	992
Шаблон (View)	993
Компоненты (Components)	996
Добавление записи	996
Показ записей	997
Показ новостей	997
Проверка корректности входных данных	998
Полномочия Компонентов	999
Достоинства подхода	1000
Система Smarty	1000
Трансляция в код на PHP	1000
Использование Smarty в MVC-схеме	1002
Инструкции Smarty	1003
Одиночные и парные теги	1003
Вставка значения переменной: <code>{<i>\$variable</i> ...}</code>	1004
Модификаторы	1004
Перебор массива: <code>{foreach}...{/foreach}</code>	1005
Ветвление: <code>{if}...{else}...{/if}</code>	1005
Вставка содержимого внешнего файла: <code>{include}</code>	1006
Вывод отладочной консоли: <code>{debug}</code>	1006
Удаление пробелов: <code>{strip}...{/strip}</code>	1006
Оператор присваивания: <code>{assign}</code>	1007
Оператор перехвата блока: <code>{capture}</code>	1007
Циклическая подстановка: <code>{cycle}</code>	1007
Глоссарий	1008
Резюме	1010
Глава 47. Динамическая загрузка данных (AJAX)	1011
Web 2.0	1011
Что такое AJAX?	1012
Google Suggest	1012
Компоненты AJAX-приложения	1013

Язык программирования JavaScript	1014
Программа, встроенная в атрибут	1014
Программа, встроенная в страницу	1014
Динамическое присваивание атрибутов, замыкания	1015
Библиотека XMLHttpRequest: кроссбраузерный AJAX и загрузка файлов	1016
Пример использования: автоподсказка набора	1018
Клиентская часть	1018
Серверная часть	1021
Загрузка файлов методом AJAX	1023
Тестирование клиентской части	1024
Тестирование серверной части	1026
Перехват ошибок в PHP-загрузчике	1027
Отправка формы целиком	1028
Асинхронность запросов	1029
Работа со встроенным объектом XMLHttpRequest	1030
Формат JSON: решение части проблем	1032
Текстовый формат и XMLHttpRequest	1033
Совместимость XMLHttpRequest и XMLHttpRequest	1033
Ссылки	1034
Резюме	1035
 Глава 48. DbSimple: упрощенный интерфейс работы с СУБД	1036
Что необходимо сценарию от СУБД?	1037
Недостатки PEAR DB, ADOdb и PDO	1038
Основные характеристики DbSimple	1039
DSN-подключение к БД	1040
Обработка ошибок	1040
Пример ошибки подключения	1041
Ошибки и исключения	1042
Основные placeholder-заполнители	1043
Строковый (бинарный) placeholder-заполнитель ?	1043
Списочный/ассоциативный placeholder-заполнитель ?a	1044
Дополнительные placeholder-заполнители	1044
Префиксный placeholder-заполнитель ?_	1044
Идентификаторный placeholder-заполнитель ?#	1045
Идентификаторно-списочный placeholder-заполнитель ?#	1045
Целочисленный placeholder-заполнитель ?d	1046
Вещественный (дробный) placeholder-заполнитель ?f	1046
Ссылочный placeholder-заполнитель ?n	1046
"Родные" placeholder-заполнители СУБД	1047
Выполнение запросов к БД	1048
Выборка всего результата: select()	1048
Выборка ассоциативного массива	1048
Выборка многомерного массива	1049
Выборка связанного дерева	1050
Выборка строки: selectRow()	1050
Выборка ячейки: selectCell()	1050
Выборка столбца: selectCol()	1051
Выборка страницы: selectPage()	1051
Выполнение обновлений: query()	1052
Обработка ошибок в запросах	1052
Макроподстановки в SQL-запросах	1053
Оптимизация связки "подготовка" + "выполнение"	1055

Журналирование запросов.....	1056
Транзакции	1057
Запросы с атрибутами.....	1058
Атрибут <i>BLOB_OBF</i> : объектные BLOB-поля	1058
Атрибут <i>CACHE</i> : кэширование запросов	1058
Зависимость от источников данных	1059
Использование <i>Cache_Lite</i> из <i>PEAR</i>	1059
Работа с кэш-хранилищем.....	1060
Ссылки	1060
Резюме.....	1061
Предметный указатель	1062

ГЛАВА 1



Принципы работы Интернета

Сеть Интернет представляет собой множество компьютеров, соединенных друг с другом кабелями, а также радиоканалами, спутниковыми каналами и т. д. Однако, как известно, одних проводов или радиоволн для передачи информации недостаточно: передающей и принимающей сторонам необходимо придерживаться ряда соглашений, позволяющих строго регламентировать передачу данных и гарантировать, что эта передача пройдет без искажений. Такой набор правил называется *протоколом передачи*. Упрощенно, протокол — это набор правил, который позволяет системам, взаимодействующим в рамках сети, обмениваться данными в наиболее удобной для них форме. Следуя сложившейся в подобного рода книгах традиции, мы вкратце расскажем, что же из себя представляют основные протоколы, используемые в Интернете.

Замечание

Иногда мы будем называть Интернет "Сетью" с большой буквы, в отличие от "сети" с маленькой буквы, которой обозначается вообще любая сеть, локальная или глобальная. Тут ситуация сходна со словом "галактика": нашу галактику часто называют Галактикой с прописной буквы, а "галактика" со строчной буквы соответствует любой другой звездной системе. На самом деле, сходство Сети и Галактики идет несколько дальше орфографии, и, думаем, вы скоро также проникнетесь этой мыслью.

Протоколы передачи данных

Необходимость некоторой стандартизации возникла чуть ли не с самого момента возникновения компьютерных сетей. Действительно, подчас одной сетью объединены компьютеры, работающие под управлением не только различных операционных систем, но нередко имеющие и совершенно различную архитектуру процессора, организацию памяти и т. д. Именно для того, чтобы обеспечивать возможность передачи между такими компьютерами, и предназначены всевозможные протоколы. Давайте рассмотрим этот вопрос чуть подробнее.

Разумеется, для разных целей существуют различные протоколы. К счастью, нам не нужно иметь представление о каждом из них — достаточно знать только тот, который мы будем использовать в Web-программировании. Таковым для нас является *протокол TCP* (Transmission Control Protocol, Протокол управления передачей дан-

ных), а точнее, *протокол HTTP* (Hypertext Transfer Protocol, Протокол передачи гипертекста), базирующийся на TCP. Протокол HTTP как раз и задействуется браузерами и Web-серверами.

Заметьте, что один протокол может использовать в своей работе другой. В мире Интернета эта ситуация является совершенно обычной. Чаще всего каждый из протоколов, участвующих в передаче данных по сети, реализуется в виде отдельного и по возможности независимого программного обеспечения или драйвера. Среди них существует некоторая иерархия, когда один протокол является всего лишь "надстройкой" над другим, тот, в свою очередь — над третьим, и т. д. до самого "низкоуровневого" драйвера, работающего уже непосредственно на физическом уровне с сетевыми картами или модемами. На рис. 1.1 приведена примерная схема процесса, происходящего при отправке запроса браузером пользователя на некоторый Web-сервер в Интернете. Прямоугольниками обозначены программные компоненты: драйверы протоколов и программы-абоненты (последние выделены жирным шрифтом), направление передачи данных указано стрелками. Конечно, в действительности процесс гораздо сложнее, но нам сейчас нет необходимости на этом останавливаться.

Обратите внимание, что в пределах каждой системы протоколы на схеме расположены в виде "стопки", один над другим. Такая структура обуславливает то, что часто семейство протоколов обмена данными в Интернете называют *стеком TCP/IP* (стек в переводе с английского как раз и обозначает "стопку").

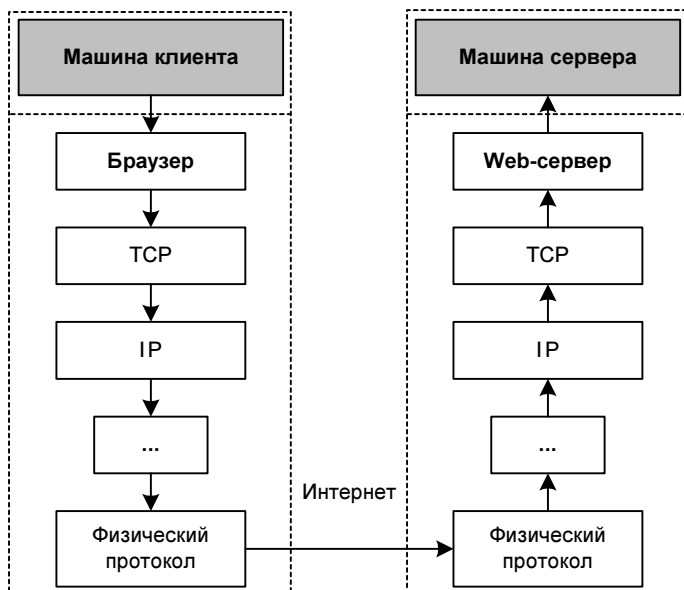


Рис. 1.1. Организация обмена данными в Интернете

Каждый из протоколов в идеале "ничего не знает" о том, какой протокол "стоит над ним". Скажем, протокол IP (который обеспечивает несколько более простой сервис

по сравнению с TCP, например, не гарантирует доставку сообщения адресату) не использует возможности протокола TCP, а TCP, в свою очередь, "не догадывается" о существовании протокола HTTP (именно его задействует браузер и понимает Web-сервер; на схеме протокол HTTP не обозначен).

Применение такой организации позволяет заметно упростить ту часть операционной системы, которая отвечает за поддержку работы с сетью. Но не пугайтесь. Нас будет интересовать в конечном итоге всего лишь протокол самого высокого уровня, "высшающийся" над всеми остальными протоколами, т. е. HTTP и то, как он взаимодействует с протоколом TCP.

Семейство TCP/IP

Вот уже несколько десятков лет основной протокол Интернета — TCP. Как часто бывает, этот выбор обусловлен скорее историческими причинами, нежели действительными преимуществами протокола (впрочем, преимуществ у TCP также предостаточно). Он ни в коей мере не претендует на роль низкоуровневого — наоборот, в свою работу TCP вовлекает другие протоколы, например, IP (в свою очередь, IP также базируется на услугах, предоставляемых некоторыми другими протоколами). Протоколы TCP и IP настолько сильно связаны, что принято объединять их в одну группу под названием *семейство TCP/IP* (в него включается также протокол UDP, рассмотрение которого выходит за рамки этой книги). Приведем основные особенности протокола TCP, входящего в семейство.

- ❑ Корректная доставка данных до места назначения гарантируется — разумеется, если такая доставка вообще возможна. Даже если связь не вполне надежна (например, на линии помехи оттого, что в кабель попала вода, замерзшая зимой и разорвавшая оболочку провода), "потерянные" фрагменты данных посылаются снова и снова до тех пор, пока вся информация не будет передана.
- ❑ Передаваемая информация представлена в виде потока — наподобие того, как осуществляется обмен с файлами практически во всех операционных системах. Иными словами, мы можем "открыть" соединение и затем выполнять с ним те же самые операции, к каким привыкли при работе с файлами. Таким образом, программы на разных машинах (возможно, находящихся за тысячи километров друг от друга), подключенных к Интернету, обмениваются данными так же непринужденно, как и расположенные на одном компьютере.
- ❑ Протокол TCP/IP устроен так, что он способен выбрать оптимальный путь распространения сигнала между передающей и принимающей стороной, даже если сигнал проходит через сотни промежуточных компьютеров. В последнем случае система выбирает путь, по которому данные могут быть переданы за минимальное время, основываясь при этом на статистическую информацию работы сети и так называемые таблицы маршрутизации.
- ❑ При передаче данные разбиваются на фрагменты — пакеты, которые и доставляются в место назначения по отдельности. Разные пакеты вполне могут следовать различными маршрутами в Интернете (особенно если их путь пролегает через десятки серверов), но для всех них гарантирована правильная "сборка" в месте назначения (в нужном порядке). Как уже упоминалось, принимающая сторона в случае обнаружения недостатка пакета запрашивает передающую систему, чтобы

та передала его еще раз. Все это происходит незаметно для программного обеспечения, эксплуатирующего TCP/IP.

В Web-программировании нам вряд ли придется работать с TCP/IP напрямую (разве что в очень экзотических случаях) — обычно можно использовать более высокоуровневые "языки", например HTTP, служащий для обмена информацией между сервером и браузером. Собственно, этому протоколу посвящена значительная часть книги. Его мы рассмотрим подробно чуть позже. А пока поговорим еще немного о том, что касается TCP/IP, чтобы не возвращаться к этому впоследствии.

Адресация в Сети

Машин в Интернете много, а будет — еще больше. Так что вопрос о том, как можно их эффективно идентифицировать в пределах всей сети, оказывается далеко не праздным. Кроме того, практически все современные операционные системы работают в многозадачном режиме (поддерживают одновременную работу нескольких программ). Это значит, что возникает также вопрос о том, как нам идентифицировать конкретную систему или программу, желающую обмениваться данными через Сеть. Эти две задачи решаются стеком TCP/IP при помощи IP-адреса и номера порта. Давайте посмотрим, как.

IP-адрес

Любой компьютер, подключенный к Интернету и желающий обмениваться информацией со своими "сородичами", должен иметь некоторое уникальное имя, или *IP-адрес*. Вот уже 30 лет среднестатистический IP-адрес выглядит примерно так:

127.12.232.56

Как мы видим, это — четыре 8-разрядных числа (принадлежащих диапазону от 0 до 255 включительно), разделенные точками. Не все числа допустимы в записи IP-адреса: ряд из них используется в служебных целях (например, адрес 127.0.0.1 (его еще часто называют *localhost*) выделен для обращения к локальной машине — той, на которой был произведен запрос, а число 255 соответствует широковещательной рассылке в пределах текущей подсети). Мы не будем здесь обсуждать эти исключения детально.

Возникает вопрос: ведь компьютеров в Интернете миллионы (а скоро будут миллиарды). Как же мы, простые пользователи, запросив IP-адрес машины, в считанные секунды с ней соединяемся? Как "оно" (и что это за "оно"?) узнает, где на самом деле расположен компьютер и устанавливает с ним связь, а в случае неверного адреса адекватно на это реагирует? Вопрос актуален, поскольку машина, с которой мы собираемся связаться, вполне может находиться за океаном, и путь к ней пролегает через множество промежуточных серверов.

В деталях вопрос определения пути к адресату довольно сложен. Однако достаточно нетрудно представить себе общую картину, точнее, некоторую ее модель. Предположим, что у нас есть 1 миллиард (10^9) компьютеров (давайте зависим цифры), каждый из которых напрямую соединен с 11 (к примеру) другими через кабели. Получается такая паутина из кабелей, не так ли? Кстати, это объясняет, почему одна из наиболее популярных служб Интернета, базирующаяся на протоколе HTTP, названа *WWW* (World Wide Web, или Всемирная паутина).

Замечание

Следует заметить, что в реальных условиях, конечно же, компьютеры не соединяют друг с другом таким большим количеством каналов. Вместо этого применяются всевозможные внутренние таблицы, которые позволяют компьютеру "знать", где конкретно располагаются некоторые ближайшие его соседи. То есть любая машина в Сети имеет информацию о том, через какие узлы должен пройти сигнал, чтобы достигнуть самого близкого к ней адресата. А если не обладает этими знаниями, то получает их у ближайшего "соседа" в момент загрузки операционной системы. Разумеется, размер таких таблиц ограничен и они не могут содержать маршруты до всех машин в Интернете (хотя в самом начале развития Интернета, когда компьютеров в Сети было немного, именно так и обстояло дело). Потому-то мы и проводим аналогию с одиннадцатью соседями.

Итак, мы сидим за компьютером номер 1 и желаем соединиться с машиной example.com с некоторым IP-адресом. Мы даем нашему компьютеру запрос: выясника у своих соседей, не знают ли они чего о example.com. Он рассылает в одиннадцать сторон этот запрос (считаем, что это занимает 0,1 с, т. к. все происходит практически одновременно — размер запроса не настолько велик, чтобы сказались задержка передачи данных), и ждет, что ему ответят.

Что же происходит дальше? Нетрудно догадаться. Каждый из компьютеров окружения действует по точно такому же плану. Он спрашивает у *своих* десяти соседей, не слышали ли они чего о example.com. Это, в свою очередь, занимает еще 0,1 с. Что же мы имеем? Всего за 0,2 с проверено уже $11 \times 10 = 110$ компьютеров. Но это еще не все, ведь процесс нарастает лавинообразно. Нетрудно подсчитать, что за время порядка 1 секунды мы "разбудим" 11×10^9 машин, т. е. в 11 раз больше, чем мы имеем!

Конечно, на самом деле процесс будет идти медленнее: какие-то системы могут быть заняты и не ответят сразу. С другой стороны, мы должны иметь механизм, который бы обеспечивал, чтобы одна машина не "опрашивалась" многократно. Но все равно, согласитесь, результаты впечатляют, даже если их и придется занизить для реальных условий хоть в 100 раз.

Замечание

В действительности дело обстоит куда сложнее. Отличия от представленной схемы частично заключаются в том, что компьютеру совсем не обязательно "запрашивать" всех своих соседей — достаточно ограничиться только некоторыми из них. Для ускорения доступа все возможные IP-адреса делятся на четыре группы — так называемые адреса подсетей классов A, B, C и D. Но для нас сейчас это не представляет никакого интереса, поэтому не будем задерживаться на деталях. О TCP/IP можно написать целые тома (что и делается).

Доменное имя

И все-таки обычным людям довольно неудобно работать с IP-представлением адреса. Действительно, куда как проще запомнить символическое имя, чем набор чисел. Чтобы облегчить простым пользователям работу с Интернетом, придумали систему DNS (Domain Name System, служба имен доменов).

Замечание

Общемировая DNS представляет собой распределенную базу данных, способную преобразовать доменные имена машин в их IP-адреса. Это не так-то просто, учитывая, что

скоро Интернет будет насчитывать десятки миллионов компьютеров. Поэтому мы не будем в деталях рассматривать то, как работает служба DNS, а займемся больше практической стороной вопроса.

Итак, при использовании DNS любой компьютер в Сети может иметь не только IP-адрес, но также и символическое имя. Выглядит оно примерно так:

www.example.msu.ru

То есть это набор слов (их число произвольно), опять же разделенных точкой. Каждое такое сочетание слов называется *доменом N-го уровня* (например, **ru** — домен первого уровня, **msu.ru** — второго, **example.msu.ru** — третьего и т. д.)

Вообще говоря, полное DNS-имя выглядит немного не так: в его конце обязательно стоит точка, например:

www.example.msu.ru.

Именно такое (вообще-то, и только такое) представление является правильным, но браузеры и другие программы часто позволяют нам опускать завершающую точку. В принятой нами терминологии будем называть эту точку *доменом нулевого уровня*, или *корневым доменом*.

Примечание

Интересно, и почему так популярна в компьютерной технике точка? В именах файлов — точка. В IP- и DNS-адресе — точка. Практически во всех языках программирования для доступа к объединениям данных — тоже точка. Существуют и другие примеры. Похоже, точка прочно въелась в наши умы, и мы уже не представляем, что бы могло ее заменить...

Нужно заметить, что одному и тому же IP-адресу вполне может соответствовать сразу несколько доменных имен. Каждое из них ведет в одно и то же место — к единственному IP-адресу. Благодаря протоколу *HTTP 1.1* (мы вскоре кратко рассмотрим его особенности) Web-сервер, установленный на машине и откликающийся на какой-либо запрос, способен узнать, какое доменное имя ввел пользователь, и соответствующим образом среагировать, даже если его IP-адресу соответствует несколько доменных имен. В последнее время HTTP 1.1 применяется практически повсеместно — не то, что несколько лет назад, поэтому все больше и больше серверов используют его в качестве основного протокола для доступа к Web.

Интересен также случай, когда одному и тому же DNS-имени сопоставлены несколько разных IP-адресов. В этом случае служба DNS автоматически выбирает тот из адресов, который, по ее мнению, ближе всего расположен к клиенту, или который давно не использовался, или же наименее загружен (впрочем, последняя оценка может быть весьма и весьма субъективна). Эта возможность часто задействуется, когда Web-сервер становится очень большим (точнее, когда число его клиентов начинает превышать некоторый предел) и его приходится обслуживать сразу нескольким компьютерам. Такая схема используется, например, на сайте компании Netscape.

Как же ведется поиск по DNS-адресу? Для начала он преобразуется специальными DNS-серверами, раскиданными по всему миру, в IP-адрес. Давайте посмотрим, как это происходит. Пусть клиентом выдан запрос на определение IP-адреса машины **www.example.com.** (еще раз обратите внимание на завершающую точку — это не конец предложения). Чтобы его обработать, первым делом посылается запрос к так

называемому корневому домену (точнее, к программе — DNS-серверу, запущенному на этом домене), который имеет имя "." (на самом деле его база данных распределена по нескольким компьютерам, но для нас это сейчас несущественно). Запрос содержит команду: вернуть IP-адрес машины (точнее, IP-адрес DNS-сервера), на котором расположена информация о домене **com**. Как только IP-адрес получен, по нему происходит аналогичное обращение с просьбой определить адрес, соответствующий домену **example** внутри домена **com** внутри корневого домена ".". В конце у предпоследней машины запрашивается IP-адрес поддомена **www** в домене **example.com**.

Каждый домен "знает" все о своих поддоменах, а те, в свою очередь — о своих, т. е. система имеет некоторую иерархичность. Корневой домен, как мы уже заметили, принято называть доменом нулевого уровня, домен **com**. (в нашем примере) — первого, **example.com**. — второго уровня, ну и т. д. При изменении доменов некоторого уровня об этом должны узнать все домены, родительские по отношению к нему, для чего существуют специальные протоколы синхронизации. Нам сейчас нет нужды вникать, как они действуют. Скажем только, что распространяются сведения об изменениях не сразу, а постепенно, спустя некоторое время, задаваемое администратором DNS-сервера, и рассылкой также занимаются DNS-серверы.

Просто? Не совсем. Представьте, какое бы произошло столпотворение на корневом домене ".", если бы все запросы на получение IP-адреса проходили через него. Чтобы этого избежать, практически все машины в Сети кэшируют информацию о DNS-запросах, обращаясь к корневому домену (и доменам первого уровня — **ru**., **com**., **org**. и т. д.) лишь изредка для обновления этого кэша. Например, пусть пользователь, подключенный через модем к провайдеру, впервые соединяется с машиной **www.example.com**.. В этом случае будет передан запрос корневому домену, а затем, по цепочке, поддомену **com**, **example** и, наконец, домену **www**. Если же пользователь вновь обратится к **www.example.com**., то сервер провайдера сразу же вернет ему нужный IP-адрес, потому что он сохранил его в своем кэше запросов ранее. Подобная технология позволяет значительно снизить нагрузку на DNS-серверы в Интернете. В то же время у нее имеются и недостатки, главный из которых — вероятность получения ложных данных, например, в случае, если хост **example.com**. только что отключился или сменил свой IP-адрес. Так как кэш обновляется сравнительно редко, мы всегда можем столкнуться с такой ситуацией.

Конечно, не обязательно, чтобы все компьютеры, имеющие различные доменные имена, были разными или даже имели уникальные IP-адреса: вполне возможна ситуация, когда на одной и той же машине, на одном и том же IP-адресе располагаются сразу несколько доменных имен.

Замечание

Здесь и далее будет подразумеваться, что одной машине в Сети всегда соответствует уникальный IP-адрес, и, наоборот, для каждого IP-адреса существует своя машина, хотя это, разумеется, не соответствует действительности. Просто так получится немного короче.

Порт

Итак, мы ответили на первый поставленный вопрос — как адресовать отдельные машины в Интернете. Теперь давайте посмотрим, как нам быть с программным обеспечением, использующим Сеть для обмена данными.

До сих пор мы расценивали машины, подключенные к Интернету, как некие неделимые сущности. Так оно, в общем-то, и есть (правда, с некоторыми оговорками) с точки зрения протокола IP. Но TCP использует в своей работе несколько другие понятия. А именно, для него отдельной сущностью является *процесс* — программа, запущенная где-то на компьютере в Интернете.

Примечание

Важно понимать разницу между программой и процессом. Программа — просто исполняемый файл на диске. Процесс же — это программа, которую загрузили в память и передали ей управление. Вы можете представить, что программа — текст книги, который существует в единственном экземпляре. Процесс же — это читатель книги, и не просто вяло водящий глазами по строчкам, а вникающий в написанное и делающий свои собственные выводы. Сколько людей — столько мнений, поэтому каждый из читателей уникален и неповторим, даже если они все читают одну и ту же книгу. Точно так же и процессы одной и той же программы различаются между собой.

Именно между процессами, а не между машинами, и осуществляется обмен данными в терминах протокола TCP. Мы уже знаем, как идентифицируются отдельные компьютеры в Сети. Осталось рассмотреть, как же TCP определяет тот процесс, которому нужно доставить данные.

Пусть на некоторой системе выполняется программа (назовем ее Клиент), которая хочет через Интернет соединиться с какой-то другой программой (Сервером) на другой машине в Сети. Для этого должен выполняться ряд условий, а именно:

- ☐ программы должны "договориться" о том, как они будут друг друга идентифицировать;
- ☐ программа Сервер должна находиться в *режиме ожидания*, что сейчас к ней кто-то подключится.

Установка соединения

Остановимся на первом пункте чуть подробнее. Термин "договориться" тут не совсем уместен (примерно так же милиция "договаривается" с только что задержанным бандитом о помещении его в тюрьму). На самом деле, как только запускается программа Сервер, она говорит драйверу TCP, что собирается использовать для обмена данными с Клиентами некоторый идентификатор, или *порт*, — целое число в диапазоне от 0 до 65 535 (именно такие числа могут храниться в ячейке памяти размером 2 байта). TCP регистрирует это в своих внутренних таблицах — разумеется, только в том случае, если какая-нибудь другая программа уже не "заняла" нужный нам порт (в последнем случае происходит ошибка). Затем Сервер переходит в режим ожидания поступления запросов, приходящих на этот порт. Это означает, что любой Клиент, который собирается вступить в "диалог" с Сервером, должен знать номер его порта. В противном случае TCP-соединение невозможно: куда передавать данные, если не знаешь, к кому подключиться?

Теперь посмотрим, какие действия предпринимает Клиент. Он, как мы условились, знает:

- ☐ IP-адрес машины, на которой запущен Сервер;
- ☐ номер порта, который использует Сервер.

Как видим, этой информации вполне достаточно, поэтому Клиент посылает драйверу TCP команду на соединение с машиной, расположенной по заданному IP-адресу, с указанием нужного номера порта. Поскольку Сервер "на том конце" готов к этому, он откликается, и соединение устанавливается.

Только что было употреблено слово "откликается", означающее, что Сервер отправляет какое-то сообщение Клиенту о своей готовности к обмену данными. Но вспомним, что для TCP существует только два понятия для идентификации процесса: адрес и порт. Так куда же направлять "отклик" Сервера? Очевидно, последний должен каким-то образом узнать, какой порт будет использовать Клиент для приема сообщений от него (ведь мы знаем, что принимать данные можно, только зарезервировав для этого у TCP номер порта). Эту информацию ему как раз и предоставляет драйвер TCP на машине Клиента, который непосредственно перед установкой соединения выбирает незанятый порт из списка свободных на данный момент портов на клиентском компьютере и "присваивает" его процессу Клиент. Затем драйвер информирует Сервер о номере порта (в первом же сообщении). Собственно, это и составляет смысл такого сообщения — передать Серверу номер порта, который будет использовать Клиент.

Обмен данными

Как только обмен "приветственными" сообщениями закончен (его еще называют "тройным рукопожатием", потому что в общей сложности посылается 3 таких сообщения), между Клиентом и Сервером устанавливается *логический канал связи*. Программы могут использовать его, как обычный канал Unix (это напоминает случай файла, открытого на чтение и запись одновременно). Иными словами, Клиент может передать данные Серверу, записав их с помощью системной функции в канал, а Сервер — принять их, прочитав из канала. Впрочем, мы вернемся к этому процессу ближе к концу книги, когда будем рассматривать функцию PHP fsockopen().

Терминология

Далее в этой книге будут применяться некоторые термины, связанные с различными "сущностями" в Интернете. Чтобы не было разногласий, сразу условимся, что понимается под конкретными понятиями. Перечислим их в том порядке, в котором они идут по логике вещей, чтобы ни одно предыдущее слово не "цеплялось" за следующее. Это — порядок "от простого к сложному". Собственно, именно по данному принципу построена вся книга, которую вы держите в руках.

Сервер

Сервер — любой отдельно взятый компьютер в Интернете, который позволяет другим машинам использовать себя в качестве "посредника" при передаче данных. Также все серверы участвуют в вышеописанной "лавине" поиска компьютера по ее IP-адресу, на многих хранится какая-то информация, доступная или нет извне. Сервер — это именно машина ("железо"), а не логическая часть Сети, он может иметь несколько различных IP-адресов (не говоря уже о доменных именах), так что вполне может выглядеть из Интернета как несколько независимых систем.