

ДЕНИС КОЛИСНИЧЕНКО



PHP 5/6 и MySQL 6

Разработка Web-приложений 2-е издание



СИНТАКСИС ЯЗЫКА PHP
ОСОБЕННОСТИ НОВЕЙШЕЙ
ВЕРСИИ PHP 6
ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ НА PHP 6
ИСПОЛЬЗОВАНИЕ СЕРВЕРОВ БАЗ
ДАННЫХ MySQL 6 И SQLite
ФУНКЦИИ ДЛЯ РАБОТЫ С MySQL 6
САМЫЕ ПОЛЕЗНЫЕ PHP-ФУНКЦИИ
ШАБЛОНИЗАТОР Smarty
ТЕХНОЛОГИЯ Ajax
РАБОТА С RSS
PHP EXTENSION AND APPLICATION
REPOSITORY (PEAR)
СИСТЕМА КОНТРОЛЯ ВЕРСИЙ CVS
ТЕСТИРОВАНИЕ С ПОМОЩЬЮ
PHPUnit
СТАТИСТИКА САЙТА
СОЗДАНИЕ СИСТЕМЫ ОПРОСОВ
СОЗДАНИЕ СИСТЕМЫ ПРОДАЖИ
НЕДВИЖИМОСТИ
SSL-СОЕДИНЕНИЯ

PRO
ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ



УДК 681.3.06
ББК 32.973.26-018.2
К60

Колисниченко Д. Н.

К60 PHP 5/6 и MySQL 6. Разработка Web-приложений. —
2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 560 с.: ил.
+ CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0581-9

На практических примерах описана разработка Web-приложений на языке PHP версий 5 и 6. Большая часть кода примеров совместима с обеими версиями PHP, но особое внимание уделено новым функциям PHP 6. Даны начала программирования на PHP: установка и настройка PHP и MySQL, выбор редактора PHP-кода, основы синтаксиса и самые полезные функции PHP. Рассмотрено создание собственного движка сайта и ряда дополнительных модулей — фотогалереи, RSS-граббера, модуля для работы с MP3, модуля продажи недвижимости, гостевой книги, а также применение мощного шаблонизатора Smarty и создание простейшего собственного шаблонизатора. В качестве хранилища данных использованы два сервера — самая современная версия MySQL 6 и "суперлегкий" сервер баз данных SQLite. Показано, как с помощью технологии Ajax добиться обновления данных на странице без ее перезагрузки. Во втором издании описаны новые инструменты для создания сложных проектов PEAR, CVS и RHPUnit. Прилагаемый компакт-диск содержит дополнительные главы, все листинги из книги, а также необходимое программное обеспечение.

Для Web-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

| | |
|-------------------------|----------------------------|
| Главный редактор | <i>Екатерина Кондукова</i> |
| Зам. главного редактора | <i>Евгений Рыбаков</i> |
| Зав. редакцией | <i>Григорий Добин</i> |
| Редактор | <i>Владимир Красовский</i> |
| Компьютерная верстка | <i>Натальи Смирновой</i> |
| Корректор | <i>Наталья Першакова</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 01.04.10.
Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 45,15.
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

ISBN 978-5-9775-0581-9

© Колисниченко Д. Н., 2010
© Оформление, издательство "БХВ-Петербург", 2010

Оглавление

| | |
|---|-----------|
| ВВЕДЕНИЕ | 1 |
| Что нового во 2-м издании..... | 1 |
| Немного истории | 2 |
| PHP 6..... | 3 |
| MySQL..... | 4 |
| Поддержка читателей..... | 5 |
| | |
| ЧАСТЬ I. ТЕОРИЯ | 7 |
| | |
| РАЗДЕЛ 1. БЫСТРЫЙ СТАРТ | 9 |
| | |
| ГЛАВА 1. УСТАНОВКА НЕОБХОДИМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ | 11 |
| 1.1. Нужно ли устанавливать программное обеспечение | 11 |
| 1.2. Выбор PHP-редактора и FTP-клиента | 13 |
| 1.3. Установка Apache + PHP + MySQL в Windows | 16 |
| 1.4. Установка Apache + PHP + MySQL в Linux | 21 |
| 1.5. Несколько советов | 22 |
| | |
| ГЛАВА 2. ПЕРВАЯ PHP-ПРОГРАММА | 23 |
| 2.1. Ваша первая программа | 23 |
| 2.2. Запуск PHP-программы..... | 24 |
| 2.3. Вывод текста без <i>echo</i> | 25 |
| | |
| ГЛАВА 3. ОСНОВЫ СИНТАКСИСА PHP | 28 |
| 3.1. Переменные..... | 28 |
| 3.1.1. Правила объявления переменных. Имена переменных | 28 |
| 3.1.2. Типы данных переменных | 29 |
| 3.1.3. Булевы переменные | 31 |
| 3.1.4. Операции над переменными..... | 32 |
| 3.1.5. Ссылки | 33 |
| 3.2. Константы..... | 34 |

| | |
|--|-----------|
| 3.3. Выражения и операции | 35 |
| 3.3.1. Что такое выражение | 35 |
| 3.3.2. Арифметические операции | 36 |
| 3.3.3. Логические выражения..... | 36 |
| 3.3.4. Приоритеты операций | 37 |
| 3.3.5. Операторы эквивалентности == и === | 38 |
| 3.3.6. Операции со строками..... | 39 |
| 3.4. Условный оператор | 40 |
| 3.5. Циклы..... | 41 |
| 3.5.1. Цикл со счетчиком..... | 41 |
| 3.5.2. Цикл <i>while</i> | 42 |
| 3.5.3. Цикл <i>do-while</i> | 42 |
| 3.5.4. Принудительное завершение цикла и пропуск итерации | 43 |
| 3.6. Оператор выбора <i>switch-case</i> | 43 |
| | |
| РАЗДЕЛ 2. ПЕРЕДАЧА ПАРАМЕТРОВ PHP-ПРОГРАММАМ..... | 45 |
| | |
| ГЛАВА 4. МЕТОДЫ GET И POST..... | 47 |
| 4.1. Интерфейс CGI..... | 47 |
| 4.2. Метод GET | 49 |
| 4.3. Метод POST | 50 |
| | |
| ГЛАВА 5. ПРОТОКОЛ HTTP И ИНТЕРФЕЙС CGI | 51 |
| 5.1. Специальные переменные окружения CGI | 51 |
| 5.2. Заголовки протокола HTTP | 52 |
| 5.3. Коды ответов протокола HTTP | 54 |
| | |
| ГЛАВА 6. ПЕРЕДАЧА ПАРАМЕТРОВ ПОСРЕДСТВОМ HTML-ФОРМЫ | 55 |
| 6.1. Создание простейшей формы и ее обработка в сценарии | 55 |
| 6.2. Создание пользовательского интерфейса с помощью формы | 58 |
| 6.2.1. Ввод текста. Теги <i>INPUT</i> и <i>TEXTAREA</i> | 60 |
| 6.2.2. Зависимые и независимые переключатели | 61 |
| 6.2.3. Списки выбора | 63 |
| 6.2.4. Форма для передачи файлов | 65 |
| 6.2.5. Кнопки | 65 |
| 6.3. Проверка параметров формы..... | 66 |
| 6.3.1. Проверка корректности e-mail | 67 |
| 6.3.2. Проверка правильности номера кредитной карты | 68 |
| 6.3.3. Удаление лишних пробелов..... | 70 |

| | |
|--|------------|
| ГЛАВА 7. ЗАПОМИНАНИЕ ПАРАМЕТРОВ С ПОМОЩЬЮ СООКИЕ И СЕССИЙ | 71 |
| 7.1. Что такое Cookies и как с ними работать | 71 |
| 7.2. Механизм сессий | 74 |
| 7.2.1. Сессии и Cookies: преимущества и недостатки | 74 |
| 7.2.2. Для чего нужны сессии..... | 74 |
| 7.2.3. Как работает механизм сессий | 76 |
| 7.2.4. Обход Cookies | 78 |
| 7.3. Массивы и Cookies..... | 78 |
| ГЛАВА 8. ОТДЕЛЬНОЕ СЛОВО О ДИРЕКТИВЕ REGISTER_GLOBS | 79 |
| 8.1. Почему опасно использовать <i>register_globals</i> | 79 |
| 8.2. Если <i>register_globals</i> отключена..... | 84 |
| 8.3. PHP 6 и <i>register_globals</i> | 86 |
| РАЗДЕЛ 3. МАССИВЫ И СПИСКИ | 87 |
| ГЛАВА 9. ОСНОВНЫЕ ОПЕРАЦИИ НАД МАССИВАМИ И СПИСКАМИ..... | 89 |
| 9.1. Массив и список. Цикл <i>foreach</i> | 89 |
| 9.2. Функции <i>list()</i> и <i>array()</i> | 91 |
| 9.3. Удаление массива | 92 |
| 9.4. Слияние массивов..... | 93 |
| 9.5. Функция <i>print_r()</i> | 95 |
| ГЛАВА 10. ФУНКЦИИ СОРТИРОВКИ МАССИВОВ..... | 97 |
| 10.1. Функции для сортировки массивов | 97 |
| 10.2. Функция <i>sort()</i> — сортировка списка | 97 |
| 10.3. Функция <i>asort()</i> — сортировка массива по значениям | 98 |
| 10.4. Функция <i>ksort()</i> — сортировка по ключам..... | 99 |
| 10.5. Функции <i>array_reverse()</i> и <i>shuffle()</i> | 100 |
| 10.6. Собственная функция сортировки | 101 |
| ГЛАВА 11. ОСОБЫЕ ОПЕРАЦИИ НАД МАССИВАМИ | 103 |
| 11.1. Добавление и удаление элементов массива | 103 |
| 11.2. Упаковка переменных в массив и их извлечение..... | 105 |
| 11.3. Получение части массива..... | 106 |

| | |
|---|------------|
| РАЗДЕЛ 4. ФУНКЦИИ В PHP | 107 |
| ГЛАВА 12. ПОЛЕЗНЫЕ СТАНДАРТНЫЕ ФУНКЦИИ | 109 |
| 12.1. Генератор случайных чисел..... | 109 |
| 12.2. Дата и время | 110 |
| 12.2.1. Кратко о timestamp..... | 110 |
| 12.2.2. Функции <i>strtotime()</i> и <i>checkdate()</i> | 111 |
| 12.2.3. Вывод даты..... | 112 |
| 12.3. Математические функции..... | 114 |
| ГЛАВА 13. ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ | 115 |
| 13.1. Основные строковые функции | 115 |
| 13.2. Специальные функции замены..... | 117 |
| 13.3. Преобразование строки..... | 117 |
| 13.4. Функции преобразования кодировок..... | 119 |
| 13.5. Функции для работы с отдельными символами строки..... | 120 |
| 13.6. Функция <i>md5()</i> | 120 |
| 13.7. Функция <i>explode()</i> : выделение подстрок | 121 |
| ГЛАВА 14. РАБОТАЕМ С ФАЙЛАМИ И КАТАЛОГАМИ | 122 |
| 14.1. Права доступа в UNIX..... | 122 |
| 14.2. Чтение файла..... | 125 |
| 14.2.1. Использование функций <i>fopen()</i> и <i>fread()</i> | 125 |
| 14.2.2. Использование функции <i>file()</i> : построчное чтение файла | 128 |
| 14.2.3. Чтение всего файла: функция <i>file_get_contents()</i> | 130 |
| 14.3. Запись файла | 130 |
| 14.4. Создание временных файлов..... | 131 |
| 14.5. Работа с CSV-файлами..... | 131 |
| 14.6. Специальные функции для работы с файлами | 134 |
| 14.6.1. Функции для работы с именами файлов..... | 134 |
| 14.6.2. Работа с правами доступа | 135 |
| 14.6.3. Копирование, переименование и удаление файлов..... | 136 |
| 14.6.4. Время доступа к файлу..... | 137 |
| 14.6.5. Другие полезные функции | 138 |
| 14.7. Совместный доступ к файлу..... | 139 |
| 14.8. Функции для работы с каталогами..... | 140 |

| | |
|---|------------|
| ГЛАВА 15. ВЫВОД ГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ СРЕДСТВАМИ PHP | 142 |
| 15.1. Библиотека GD..... | 142 |
| 15.1.1. Получение информации об изображении..... | 143 |
| 15.1.2. Конвертирование графических форматов | 146 |
| 15.1.3. Вывод текста поверх картинок..... | 149 |
| 15.1.4. Прозрачность..... | 152 |
| 15.2. Изменение размера изображения..... | 152 |
| 15.3. Создание водяных знаков | 154 |
| ГЛАВА 16. РАБОТА С СЕТЕВЫМИ СОКЕТАМИ В PHP. | |
| СЕТЕВЫЕ ФУНКЦИИ | 157 |
| 16.1. Еще раз о том, что такое сокет | 157 |
| 16.2. Использование сокетов | 158 |
| 16.3. Пример использования сокетов..... | 159 |
| 16.4. Блокирующий и неблокирующий режимы сокета | 163 |
| 16.5. DNS-функции | 163 |
| ГЛАВА 17. СОБСТВЕННЫЕ ФУНКЦИИ..... | 165 |
| 17.1. Зачем нужны собственные функции | 165 |
| 17.2. Особенности функций в PHP..... | 166 |
| 17.3. Объявление функции..... | 166 |
| 17.4. Области видимости функции..... | 167 |
| 17.5. Вложенность функций..... | 168 |
| 17.6. Переменное число аргументов | 170 |
| 17.7. Передача массивов в качестве параметров | 171 |
| РАЗДЕЛ 5. СЕРВЕРЫ БАЗ ДАННЫХ MYSQL 6 И SQLITE | 173 |
| ГЛАВА 18. ОСНОВЫ SQL | 175 |
| 18.1. Немного истории | 175 |
| 18.2. Преимущества SQL..... | 176 |
| 18.3. Как выглядят запросы | 177 |
| 18.4. Что такое база данных..... | 177 |
| 18.5. Создание таблиц | 179 |
| 18.6. Добавление записей в таблицу | 183 |
| 18.7. Обновление записей | 184 |
| 18.8. Выборка записей..... | 185 |
| 18.9. Удаление записей..... | 186 |

| | |
|---|------------|
| 18.10. Встроенные функции..... | 187 |
| 18.11. Группировка записей. Сложные запросы..... | 188 |
| ГЛАВА 19. ФУНКЦИИ ДЛЯ РАБОТЫ С MYSQL..... | 192 |
| 19.1. Подключение к серверу MySQL..... | 192 |
| 19.2. Несколько MySQL-соединений..... | 194 |
| 19.3. Передача запросов серверу..... | 195 |
| 19.4. Работа с базой данных. Создание базы данных..... | 200 |
| 19.5. Функция <i>mysql_real_escape_string(\$content)</i> | 200 |
| ГЛАВА 20. АЛЬТЕРНАТИВНАЯ БАЗА ДАННЫХ SQLITE..... | 202 |
| 20.1. MySQL vs SQLite: что лучше..... | 202 |
| 20.2. Открытие базы данных..... | 204 |
| 20.3. Передача запросов..... | 205 |
| 20.4. Работа с результатом запроса..... | 205 |
| 20.5. Список PHP-функций для работы с SQLite..... | 207 |
| ГЛАВА 21. ПОЛЕЗНЫЕ ПРИЕМЫ ПРИ РАБОТЕ С MYSQL 6..... | 210 |
| 21.1. Выбор кодировки..... | 210 |
| 21.2. Сортировка: вывод новинок. Вывод случайных записей..... | 211 |
| 21.3. Постраничный вывод таблицы..... | 212 |
| РАЗДЕЛ 6. ИНСТРУМЕНТЫ ДЛЯ СОЗДАНИЯ СЛОЖНЫХ ПРОЕКТОВ..... | 219 |
| ГЛАВА 22. РАЗРАБОТКА СОБСТВЕННОГО ШАБЛОНИЗАТОРА..... | 221 |
| 22.1. Организация файлов и каталогов проекта..... | 221 |
| 22.2. Выносим параметры в отдельный файл..... | 224 |
| 22.3. Подключение дополнительных файлов..... | 225 |
| 22.3.1. Инструкции <i>include</i> и <i>require</i> | 225 |
| 22.3.2. Альтернативный способ подключения сценариев..... | 226 |
| 22.3.3. Инструкции <i>include_once</i> и <i>require_once</i> | 228 |
| 22.4. Шаблоны..... | 228 |
| ГЛАВА 23. ШАБЛОНИЗАТОР SMARTY..... | 233 |
| 23.1. Что такое Smarty..... | 233 |
| 23.2. Установка Smarty..... | 234 |

| | |
|---|-----|
| 23.3. Создание <code>setup.php</code> | 237 |
| 23.4. Разработка шаблонов Smarty | 238 |
| 23.4.1. Комментарии в шаблонах | 238 |
| 23.4.2. Переменные в Smarty | 239 |
| 23.4.3. Файлы конфигурации шаблонов | 240 |
| 23.4.4. Служебная переменная <code>{<i>\$smarty</i>}</code> | 241 |
| 23.4.5. Модификаторы переменных | 243 |
| 23.4.6. Стандартные (встроенные) функции Smarty | 246 |
| Функции <code>{include}</code> и <code>{insert}</code> | 247 |
| Функция <code>{foreach}</code> | 247 |
| Функции <code>{if}</code> , <code>{elseif}</code> , <code>{else}</code> | 249 |
| Функция <code>{capture}</code> | 251 |
| Функция <code>{php}</code> | 251 |
| Функция <code>{strip}</code> | 251 |
| 23.4.7. Пользовательские функции Smarty | 252 |
| Функция <code>{assign}</code> | 252 |
| Функция <code>{cycle}</code> | 252 |
| Функция <code>{fetch}</code> | 253 |
| Функции <code>{html_checkboxes}</code> и <code>{html_radios}</code> | 253 |
| Функция <code>{html_image}</code> | 254 |
| Функция <code>{html_select_date}</code> | 254 |
| Функция <code>{html_select_time}</code> | 257 |
| Функция <code>{html_table}</code> | 257 |
| 23.5. Smarty для программиста | 258 |
| 23.5.1. Специальные переменные | 258 |
| 23.5.2. Полезные методы класса <i>Smarty</i> | 261 |

Глава 24. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ262

| | |
|---|-----|
| 24.1. Основы ООП | 262 |
| 24.2. Классы и объекты | 263 |
| 24.3. Конструкторы и деструкторы класса | 266 |
| 24.4. Наследование классов. Полиморфизм | 267 |
| 24.5. Новые возможности PHP 5/6 | 268 |
| 24.5.1. Область видимости членов класса | 268 |
| 24.5.2. Абстрактные классы и методы | 269 |
| 24.5.3. Служебное слово <i>final</i> | 270 |
| 24.5.4. Клонирование объектов | 271 |
| 24.5.5. Обработка исключительных ситуаций | 272 |
| 24.5.6. Константы-члены класса | 273 |
| 24.5.7. Статические члены класса | 274 |

| | |
|--|------------|
| 24.5.8. Оператор <i>instanceof</i> | 274 |
| 24.5.9. Итераторы..... | 275 |
| 24.5.10. Пространства имен..... | 275 |
| ГЛАВА 25. МЕХАНИЗМ СЕССИЙ..... | 276 |
| 25.1. Для чего нужны сессии..... | 276 |
| 25.2. Как работает механизм сессий..... | 278 |
| 25.3. Обход Cookies..... | 279 |
| 25.4. Сценарий аутентификации..... | 280 |
| ГЛАВА 26. ВВЕДЕНИЕ В PEAR..... | 285 |
| 26.1. Серьезные проекты и PEAR..... | 285 |
| 26.2. Пример использования класса <i>DB</i> | 287 |
| ГЛАВА 27. КОНТРОЛЬ ВЕРСИЙ..... | 291 |
| 27.1. Выбор системы контроля версий..... | 291 |
| 27.2. Практическое использование TortoiseHG (Mercurial)..... | 293 |
| 27.3. Просмотр внесенных изменений..... | 295 |
| ГЛАВА 28. ТЕСТИРОВАНИЕ РНР-СЦЕНАРИЕВ..... | 298 |
| 28.1. Программа работает, но не так, как нам нужно..... | 298 |
| 28.2. "Самодельные" точки останова..... | 300 |
| 28.3. Система автоматического тестирования..... | 300 |
| 28.4. Директива <i>error_reporting</i> | 305 |
| ЧАСТЬ II. ПРАКТИКА..... | 307 |
| РАЗДЕЛ 7. РАЗРАБОТКА ОСНОВНЫХ ЭЛЕМЕНТОВ САЙТА..... | 309 |
| ГЛАВА 29. ЗАГРУЗКА ФАЙЛОВ НА СЕРВЕР..... | 311 |
| 29.1. Что нужно знать о загрузке файлов на сервер..... | 311 |
| 29.2. Реализация загрузки файла..... | 314 |
| 29.3. Загрузка нескольких файлов..... | 317 |
| 29.4. Проблемы при загрузке файлов..... | 319 |

| | |
|---|------------|
| ГЛАВА 30. ИСПОЛЬЗОВАНИЕ FTP-ФУНКЦИЙ..... | 321 |
| 30.1. Функции для работы с FTP | 321 |
| 30.2. Примеры использования FTP-функций | 325 |
| ГЛАВА 31. ОТПРАВКА И ПРИЕМ ПОЧТЫ | 328 |
| 31.1. Отправка почты средствами PHP — функция <i>mail()</i> | 328 |
| 31.2. Отправка писем с вложениями — класс <i>HtmlMimeMail</i> | 330 |
| 31.2.1. Отправка сообщения..... | 330 |
| 31.2.2. Проблемы при отправке сообщения | 334 |
| 31.2.3. MIME-типы..... | 336 |
| 31.3. Получение писем по протоколу POP3 | 337 |
| ГЛАВА 32. РАБОТА С RSS: ПОЛУЧАЕМ НОВОСТИ НА САЙТ | 342 |
| 32.1. Краткие сведения о RSS | 342 |
| 32.2. Формат RSS-файла | 343 |
| 32.3. Написание сценария импорта новостей | 345 |
| 32.4. Подключение файла <i>import.php</i> к сайту..... | 349 |
| 32.5. Создание граббера новостей..... | 349 |
| ГЛАВА 33. ПОИСК С ИСПОЛЬЗОВАНИЕМ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ..... | 354 |
| 33.1. Нужно что-то найти... .. | 354 |
| 33.2. Язык регулярных выражений RegEx | 355 |
| 33.3. Управляющие конструкции | 357 |
| 33.3.1. Квантификаторы | 357 |
| 33.3.2. Альтернативный оператор | 358 |
| 33.3.3. Скобки..... | 358 |
| 33.4. Псевдосимволы..... | 359 |
| 33.5. Практическое использование RegEx-функций | 359 |
| ГЛАВА 34. РАБОТАЕМ С MP3 | 364 |
| 34.1. Формат MP3 | 364 |
| 34.2. Библиотека PEAR | 365 |
| 34.3. Вывод ID3-тегов..... | 366 |
| 34.4. Редактирование ID3-тегов | 368 |
| 34.5. Удаление тега | 369 |
| ГЛАВА 35. ТЕХНОЛОГИЯ AJAX..... | 370 |
| 35.1. Что такое Ajax | 370 |
| 35.2. Ваше первое Ajax-приложение..... | 371 |

| | |
|--|------------|
| РАЗДЕЛ 8. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ САЙТА | 377 |
| ГЛАВА 36. СОЗДАНИЕ ПРОСТЕЙШЕГО ДВИЖКА САЙТА | 379 |
| 36.1. Планирование движка | 379 |
| 36.1.1. Зачем нужно разрабатывать собственный движок | 379 |
| 36.1.2. Необходимые нам функции движка | 381 |
| 36.1.3. "Принципиальная схема" движка | 383 |
| 36.2. Основные функции движка | 385 |
| 36.2.1. Разработка TPL-шаблона | 385 |
| 36.2.2. Файл настроек | 386 |
| 36.2.3. Основной файл CMS — index.php | 386 |
| 36.2.4. Проектирование базы данных | 389 |
| Таблица static | 390 |
| Таблица cats | 390 |
| Таблица pages | 392 |
| 36.2.5. Иерархическая структура сайта | 393 |
| Алгоритм работы меню | 393 |
| Сценарий menu.php | 395 |
| Вывод содержимого раздела и страницы | 402 |
| 36.3. Дополнительные функции движка | 406 |
| 36.3.1. Вывод информации из таблицы static | 406 |
| 36.3.2. Функция вывода содержимого HTML-файла | 408 |
| 36.3.3. Версия для печати | 409 |
| 36.4. Где взять листинги этой главы | 410 |
| ГЛАВА 37. СОЗДАНИЕ ФОТОГАЛЕРЕИ | 412 |
| 37.1. Постановка задачи | 412 |
| 37.2. Загрузка изображений на сервер | 412 |
| 37.3. Вывод галереи | 417 |
| ГЛАВА 38. ГОСТЕВАЯ КНИГА | 420 |
| 38.1. Пережиток прошлого? | 420 |
| 38.2. Разработка базы данных и структура гостевой книги | 421 |
| 38.3. Вывод гостевой книги | 421 |
| 38.4. Добавление записей в гостевую книгу | 425 |
| 38.5. Сервисный сценарий gb_service.php | 429 |
| ГЛАВА 39. ИНТЕГРАЦИЯ ГАЛЕРЕИ LIVEJOURNAL И ВАШЕГО САЙТА | 431 |
| 39.1. Что такое Живой журнал | 431 |
| 39.2. Интеграция фотогалереи LiveJournal и сайта | 433 |
| 39.3. Настройка внешнего вида галереи | 437 |

| | |
|---|------------|
| ГЛАВА 40. СОЗДАНИЕ СЧЕТЧИКА САЙТА | 439 |
| 40.1. Постановка задачи | 439 |
| 40.2. Файл конфигурации..... | 440 |
| 40.3. Разработка таблиц counter и ipaddr | 440 |
| 40.4. Сценарий counter.php..... | 442 |
| 40.5. Сценарий reset_counter.php | 447 |
| ГЛАВА 41. СТАТИСТИКА САЙТА..... | 448 |
| 41.1. Методы сбора статистики | 448 |
| 41.2. Программы-анализаторы журналов Web-сервера | 450 |
| 41.3. Системы статистики | 451 |
| ГЛАВА 42. ГОЛОСОВАНИЯ (ОПРОСЫ)..... | 455 |
| 42.1. Разработка собственной системы голосования | 455 |
| 42.2. Разработка сценария poll_form.php | 457 |
| 42.3. Сценарий poll_process.php | 459 |
| 42.4. Сценарий poll_results.php | 461 |
| РАЗДЕЛ 9. СЛОЖНЫЙ ПРОЕКТ: САЙТ ПО ПРОДАЖЕ НЕДВИЖИМОСТИ | 465 |
| ГЛАВА 43. ПОСТАНОВКА ЗАДАЧИ | 467 |
| 43.1. Функции будущего проекта..... | 467 |
| 43.2. Разработка базы данных..... | 468 |
| ГЛАВА 44. РАЗРАБОТКА ОСНОВНОЙ ЧАСТИ САЙТА | 479 |
| 44.1. С чего начать | 479 |
| 44.2. Реализация основных функций системы..... | 481 |
| 44.2.1. Аутентификация пользователей | 481 |
| 44.2.2. Вывод VIP-объявлений | 485 |
| 44.2.3. Вывод объявлений об услугах | 488 |
| 44.2.4. Вывод рекламных баннеров..... | 489 |
| 44.2.5. Постраничный вывод объявлений | 489 |
| 44.3. На что обратить внимание | 493 |
| ГЛАВА 45. РАЗРАБОТКА ПАНЕЛИ АДМИНИСТРИРОВАНИЯ | 501 |
| 45.1. Функции панели управления | 501 |
| 45.2. Управление VIP-объявлениями..... | 502 |

| | |
|--|------------|
| 45.3. Массовая отправка электронного сообщения..... | 503 |
| 45.4. Общие операции с базой данных | 504 |
| 45.5. Редактирование статей, новостей и контактов | 505 |
| РАЗДЕЛ 10. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ САЙТА..... | 507 |
| ГЛАВА 46. SSL-СОЕДИНЕНИЯ..... | 509 |
| 46.1. Защищаем передаваемые данные..... | 509 |
| 46.2. Настройка SSL в DirectAdmin..... | 510 |
| 46.3. SSL-переменные..... | 512 |
| ГЛАВА 47. ЗАЩИТА PHP С ПОМОЩЬЮ КОНФИГУРАЦИОННОГО ФАЙЛА | 515 |
| 47.1. Конфигурационный файл php.ini | 515 |
| 47.2. Отключение потенциально опасных функций..... | 517 |
| 47.3. Рекомендованные значения некоторых конфигурационных директив | 517 |
| ГЛАВА 48. ЗАЩИТА САЙТА ОТ АТАК..... | 519 |
| 48.1. Сайт в опасности..... | 519 |
| 48.2. Два самых распространенных метода взлома..... | 520 |
| 48.3. Межсайтовый скриптинг | 521 |
| 48.4. SQL-инъекции | 523 |
| 48.5. Флуд | 527 |
| 48.6. Защита форума PHPBB2 от спамеров | 528 |
| ПРИЛОЖЕНИЕ. ОПИСАНИЕ CD..... | 531 |
| ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ | 533 |

ГЛАВА 3



Основы синтаксиса PHP

3.1. Переменные

3.1.1. Правила объявления переменных.

Имена переменных

Если вы писали до этого на других языках программирования, то наверняка знакомы с понятием переменной, которое присутствует и в PHP. Говоря формально, переменная — это поименованная область памяти. Области памяти присваивается выбранное программистом имя, по которому потом программа обращается к данным. Все данные, с которыми вы будете работать, будут храниться в переменных.

В других языках программирования переменные нужно объявлять до их первого использования, например в языке Pascal:

```
var  
    I : integer;
```

PHP не требует явного объявления переменных. Но если переменные не объявлены, как тогда PHP знает, что есть переменная, а что — обычный символ? Для выделения переменных из кода программы PHP требует, чтобы перед именем переменной указывался символ `$`. Например: `$i`, `$string`.

То, что PHP не требует объявления переменной, не освобождает вас от ее инициализации, то есть от задания переменной начального значения. Инициализация не обязательна, но весьма желательна. Предположим, что вы используете переменную `$i` как счетчик с шагом 2, то есть после определенного события, например после каждой итерации цикла, выполняется следующий оператор:

```
$i = $i + 2;
```

Если вы забыли присвоить переменной `$i` начальное значение до выполнения этого оператора, результат будет непредсказуемым. Поэтому возьмите за правило инициализировать каждую переменную, которую вы собираетесь использовать, например:

```
$i = 0;
```

Вернемся к именам переменных. Как уже было сказано, перед именем переменной нужно обязательно указывать символ доллара. Например:

```
i = $i + 2;           // этот код неправильный!  
i = i + 2;           // этот код неправильный!  
$i = $i + 2;        // этот код ПРАВИЛЬНЫЙ
```

Имя переменной может состоять из латинских символов, знака подчеркивания и цифр, причем не может начинаться с цифры.

Имена переменных чувствительны к регистру символов, то есть `$a` и `$A` — две разные переменные:

```
$a = 1;  
$A = 2;
```

```
echo $a;  
echo $A;
```

Наш код выведет, как и следовало ожидать, 12. Если бы переменные не были чувствительны к регистру символов, код вывел бы 11.

3.1.2. Типы данных переменных

Обратите внимание, что при объявлении переменной на языке Pascal были указаны имя переменной и ее тип. PHP не требует явного объявления типа переменной, но это не означает, что в PHP нет типов переменных. Просто PHP по-особому работает с переменными.

Предположим, что вы объявили две переменные. Одна переменная (`$a`) содержит число, а вторая переменная (`$b`) — массив. Присвоим второй переменной значение первой переменной:

```
$b = $a;
```

Вы только задумайтесь, что происходит. Вы присваиваете число массиву! В любом другом языке программирования вы бы получили ошибку несовместимости типов переменных, но не в PHP. PHP возьмет да и скопирует в

переменную `$b` не только значение переменной `$a`, но и тип (структуру) переменной. Да, в результате массив будет потерян, `$b` станет обычной целой переменной и будет содержать значение переменной `$a`.

Такая "молчаливость" PHP довольно удобна, но с другой стороны, если вы невнимательны, она может стать причиной трудно отслеживаемых ошибок.

Вот основные типы переменных в PHP:

- ❑ `int` — целой переменной вы можете присвоить значения от $-2\,147\,483\,648$ до $2\,147\,483\,647$;
- ❑ `double` — переменная, содержащая вещественное число большой точности;
- ❑ `string` — наверное, самый важный тип данных в PHP, поскольку очень много операций связано именно с обработкой строк;
- ❑ `array` — массивам и спискам посвящен целый раздел книги;
- ❑ `object` — PHP является объектно-ориентированным языком программирования, что, впрочем, требуется далеко не всегда — вы можете написать превосходный сценарий, работающий и без объектов;
- ❑ `bool` — логическая переменная (может принимать значения `true` или `false`);
- ❑ `link` — ссылка, особый тип, о котором мы поговорим отдельно.

Для проверки типа переменной используются функции, приведенные в табл. 3.1.

Таблица 3.1. Функции проверки типа переменной

| Функция | Описание |
|-----------------------------|---|
| <code>is_int(\$x)</code> | Возвращает <code>true</code> , если указанная переменная — целое число |
| <code>is_double(\$x)</code> | Возвращает <code>true</code> , если указанная переменная — вещественное число |
| <code>is_string(\$x)</code> | Проверяет, является ли переменная строкой |
| <code>is_array(\$x)</code> | Проверяет, является ли переменная массивом |
| <code>is_object(\$x)</code> | Возвращает <code>true</code> , если указанная переменная — объект |
| <code>is_bool(\$x)</code> | Проверяет, является ли переменная логической переменной |

Таблица 3.1 (окончание)

| Функция | Описание |
|--------------|---|
| gettype(\$x) | <p>Возвращает строку, которая описывает тип переменной:</p> <ul style="list-style-type: none"> <input type="checkbox"/> "boolean" — логическое значение; <input type="checkbox"/> "integer" — целое значение; <input type="checkbox"/> "double" — вещественное значение; <input type="checkbox"/> "string" — строка; <input type="checkbox"/> "array" — массив; <input type="checkbox"/> "object" — объект; <input type="checkbox"/> "resource" — ресурс (например, результат обработки MySQL-запроса); <input type="checkbox"/> "NULL" — переменная не имеет значения (не инициализирована), следовательно, невозможно установить ее тип; <input type="checkbox"/> "unknown type" — неизвестный тип (тип определить не удалось) |

3.1.3. Булевы переменные

Отдельного разговора заслуживают логические переменные. Мы привыкли, что в других языках программирования значения `true` и `false` ассоциируются соответственно со значениями 1 и 0. Но в PHP значение `false` ассоциируется с пустой строкой, а `true` — с любым ненулевым значением, например:

```
$a = 10001;
if ($a == true) echo 'Переменная $a — истинна';
```

ПРИМЕЧАНИЕ

Обратите внимание, что в операторе `echo` используются одинарные кавычки, а не двойные.

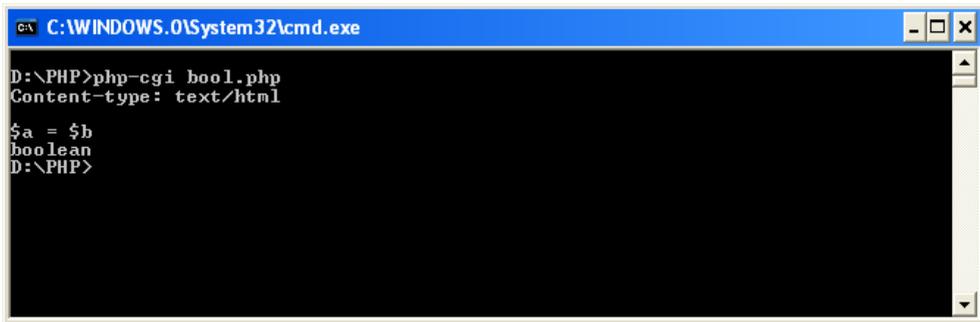
Поскольку со значением `true` ассоциируется любое ненулевое значение, то мы увидим следующую строку:

```
Переменная $a — истинна
```

Рассмотрим еще один пример:

```
$a = 1001; $b = true;  
if ($a == $b) echo '$a = $b'; // сравниваем переменные  
echo "\n".gettype($b); // выводим тип переменной $b
```

Самое интересное, что наш сценарий сообщит, что переменные `$a` и `$b` равны (рис. 3.1). Это происходит потому, что переменной `$b` присвоен логический тип (это так и есть), и PHP считает, что и переменная `$a` тоже логического типа.



```
C:\WINDOWS.0\System32\cmd.exe  
D:\PHP>php-cgi bool.php  
Content-type: text/html  
$a = $b  
boolean  
D:\PHP>
```

Рис. 3.1. Сравнение логической и целой переменных

3.1.4. Операции над переменными

Над переменной вы можете выполнить всего три операции:

- присвоить значение переменной — константу, результат выполнения функции или выражения:

```
$x = 1;  
$x = pi;  
$x = func($y);  
$x = ($b * 2) / 100;
```

- уничтожить переменную — для этого используется функция `unset()`:

```
unset($a);
```

- проверить существование переменной — используется функция `isset()`:

```
if (isset($x)) echo 'Переменная $x установлена';
```

3.1.5. Ссылки

Если вы знакомы с программированием на С, то вы знакомы с указателями. В PHP нет указателей, зато есть ссылки. Ссылки можно расценивать как псевдонимы переменных. Предположим, что у вас есть переменная-массив `$arr1` объемом 1 Мбайт. Если вы присвоите ее значение другой переменной, `$arr2`, то у вас будет два одинаковых массива общим размером 2 Мбайт.

Ссылки позволяют обращаться к одним и тем же данным под разными именами:

```
$a = 100;  
$link_on_a = &$a;
```

```
echo $link_on_a;
```

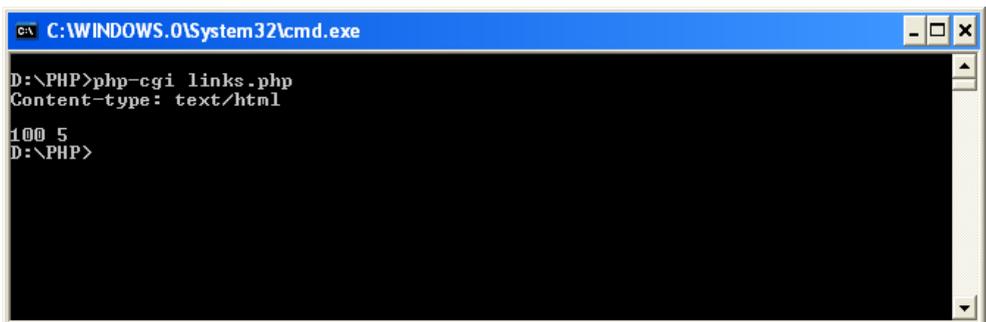
```
echo " ";      // пробел
```

```
$link_on_a = 5;  
echo $a;
```

Сначала мы объявляем переменную `$a`. Затем мы объявляем ссылку `$link_on_a` на переменную `$a`. Синтаксис объявления ссылки следующий:

```
ссылка = &переменная;
```

После чего мы выводим переменную `$link_on_a`. Она равна 100, как и следовало ожидать. Затем мы присваиваем ссылке `$link_on_a` значение 5 и выводим значение переменной `$a`. Сценарий выведет значение 5 (рис. 3.2).



```
C:\WINDOWS.0\System32\cmd.exe  
D:\PHP>php-cgi links.php  
Content-type: text/html  
100 5  
D:\PHP>
```

Рис. 3.2. Пример использования ссылок

Приведенные ссылки называются *жесткими*. Существуют еще "мягкие" (символические) ссылки. Символическая ссылка — это не псевдоним переменной, а переменная, содержащая имя другой переменной.

Работать с мягкими ссылками нужно следующим образом. Сначала нужно создать переменную, на которую будет указывать ссылка. Затем нужно другой переменной присвоить имя первой переменной. Чтобы создать мягкую ссылку, нужно обратиться ко второй переменной так:

```
$$переменная
```

Рассмотрим следующий фрагмент кода:

```
$a = 100;                // исходная переменная
$soft_link = "a";       // эта переменная будет играть роль
                        // ссылки

echo $$soft_link;       // это уже ссылка, выведет 100

echo " ";               // выводим пробел

echo $soft_link;        // выведет просто 'a' (без кавычек)

$$soft_link = 5;       // присваиваем ссылке значение 5

echo " ";               // еще один пробел

echo $a;                // выводим переменную $a, будет выведено 5
```

3.2. Константы

Константы содержат постоянные значения, изменить которые в процессе выполнения программы вы не можете. Для объявления констант используется функция `define()`, синтаксис ее таков:

```
define(имя, значение, чувствительность_к_регистру);
```

Вот примеры объявления констант:

```
define("const", "константа", true);
define("A", "1");
```

Первый параметр функции `define()` — имя константы, второй — значение. Третий параметр особый. Если он `true`, имя константы будет чувствительно к регистру. По умолчанию константы чувствительны к регистру, то есть `CONST` и `const` — это две разные константы.

При использовании константы указывать знак доллара не нужно:

```
// выводим константы
echo const;
echo A;
```

Вы можете использовать следующие стандартные константы:

- ❑ `_FILE_` — имя текущего сценария;
- ❑ `PHP_OS` — название операционной системы, под которой выполняется PHP;
- ❑ `PHP_VERSION` — версия PHP.

3.3. Выражения и операции

3.3.1. Что такое выражение

Выражение — это часть инструкции или сама инструкция, возвращающая значение. Практически все инструкции программы являются выражениями. Даже оператор присваивания:

```
$a = 1;
```

Результат этой операции — 1, он будет присвоен переменной `$a`. Само выражение в этом случае выглядит так:

```
1
```

Выражением может быть не только значение, но и инициализированная переменная:

```
$a = 1;
$b = $a;
```

В PHP оператор присваивания может выглядеть довольно необычно:

```
$a = $b = 1;
```

Интересно, что в выражении можно даже инициализировать переменную:

```
$a = 50 * ($c = 7) * $c;
```

3.3.2. Арифметические операции

Как и в любом другом языке программирования, вы можете использовать следующие арифметические операции:

- + — сложение;
- - — вычитание;
- * — умножение;
- / — деление;
- $A \% B$ — остаток от деления A на B .

Кроме того, вы можете использовать операции инкремента и декремента:

- $\$a++$ — увеличить переменную $\$a$ на 1;
- $\$b--$ — уменьшить переменную $\$b$ на 1.

3.3.3. Логические выражения

Логическим называется выражение, результатом которого является истина или ложь. Обычно в логических выражениях используются следующие операторы сравнения:

- < — меньше;
- > — больше;
- != — не равно;
- == — равно;
- <= — меньше или равно;
- >= — больше или равно.

Также вы можете использовать следующие сугубо логические операции:

- && — бинарная операция И (AND); истинна, если оба операнда истинны;
- || — бинарная операция ИЛИ (OR); истинна, если один из операндов true;
- ! — унарная операция отрицания (NOT).

Примеры использования логических операций:

```
$a = true;  
$b = false;
```

```

$c = $a || $b;      // $c = true;
$d = $a && $b;      // $d = false;

$e = !$a;           // $e = false;

```

3.3.4. Приоритеты операций

Операции в PHP выполняются в соответствии с приоритетом выполнения операций, приведенным в табл. 3.2. Для изменения порядка выполнения операций можно использовать скобки, например:

```
$a = (2 + 2) * 2;
```

Таблица 3.2. Приоритет операций

| Приоритет | Порядок выполнения | Операторы |
|-----------|--------------------|--|
| 13 | справа налево | =, +=, . =, *=, /=, %=, >>=, <<=, &=, ^=, = |
| 12 | слева направо | |
| 11 | слева направо | && |
| 10 | слева направо | |
| 9 | слева направо | ^ |
| 8 | слева направо | & |
| 7 | слева направо | ==, != |
| 6 | слева направо | <, <=, >, >= |
| 5 | слева направо | <<, >> |
| 4 | слева направо | +, - |
| 3 | слева направо | *, /, % |
| 2 | справа налево | ++(префикс), --(префикс) |
| 1 | слева направо | (постфикс)++, (постфикс)-- |

ПРИМЕЧАНИЕ

Приоритет операций в табл. 3.2 представлен от низшего к высшему.

ПРИМЕЧАНИЕ

Операции `>>` и `<<` называются битовыми. Мы их не рассматриваем, поскольку при Web-разработке они используются очень редко, а на роль системного языка программирования (как C) PHP явно не тянет!

3.3.5. Операторы эквивалентности `==` и `===`

Для сравнения значений используется оператор равенства `==`, например:

```
if ($a == $b) ...
```

Иногда этот оператор работает не совсем так, как мы ожидаем. Рассмотрим следующий код:

```
$x = 1;
```

```
$y = 1;
```

```
$a = 0;
```

```
$b = "";
```

```
if ($x == $y) echo 'Переменные $x и $y равны';
```

```
if ($a == $b) echo 'Переменные $a и $b равны';
```

Переменные `$x` и `$y` действительно равны, поэтому мы увидим сообщение:

```
Переменные $x и $y равны
```

Но переменные `$a` и `$b` вообще не могут быть равны, они даже разных типов, однако PHP сообщает, что они равны:

```
Переменные $a и $b равны
```

Почему так происходит? Просто PHP воспринимает `$a` и `$b` как логические. А 0 и пустая строка соответствуют `false`, поэтому он и сообщает, что переменные равны.

Чтобы избежать подобной неразберихи, нужно использовать оператор сравнения `===` (три знака равенства):

```
if ($a === $b) echo 'Переменные $a и $b равны';
```

В этом случае строка 'Переменные `$a` и `$b` равны' не будет выведена.

Помните, что для оператора `===` не предусмотрен обратный оператор `!==`, вместо него вы должны использовать следующую конструкцию:

```
if (!( $a === $b )) echo '$a <> $b';
```