



Самоучитель

Максим Кузнецов, Игорь Симдянов

PHP 5/6

Третье издание



Объектно-ориентированное программирование

Работа с базами данных

Защита приложений от взлома

Актуальные примеры

Новое в PHP 5/6

УДК 681.3.06
ББК 32.973.26-018.2
К89

Кузнецов, М. В.

К89 Самоучитель PHP 5/6 / М. В. Кузнецов, И. В. Симдянов —
3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 672 с.: ил.

ISBN 978-5-9775-0409-6

Описаны самые последние версии языка разработки серверных сценариев PHP — 5.3 и 6.0. Рассмотрены основы языка, вопросы объектно-ориентированного программирования на PHP, обработки исключительных ситуаций, взаимодействия с MySQL, регулярные выражения, работа с электронной почтой. Книга содержит множество примеров, взятых из реальной практики разработки динамических Web-сайтов.

Третье издание книги, ранее вышедшей под названием "Самоучитель PHP 5", существенно переработано, дополнено и будет интересно не только программистам, впервые знакомящимся с языком, но и читателям предыдущих изданий книги и профессионалам.

Для программистов и Web-разработчиков

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>Анна Кузьмина</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Наталья Першакова</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 03.12.08.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 54,18.

Тираж 2500 экз. Заказ №

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

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.002108.02.07

от 28.02.2007 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 978-5-9775-0409-6

© Кузнецов М. В., Симдянов И. В., 2008

© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

ВВЕДЕНИЕ	1
Нововведения PHP 6	2
Благодарности.....	2
ГЛАВА 1. ЧТО ПРЕДСТАВЛЯЕТ СОБОЙ PHP?	3
1.1. История PHP	3
1.2. Место и роль PHP в Интернете	5
1.2.1. Серверные технологии	6
UNIX-подобная операционная система	6
Web-сервер.....	7
Серверный язык.....	7
Файлы и базы данных	8
Электронная почта	9
1.2.2. Клиентские технологии.....	9
Web-браузеры, HTML.....	10
Каскадные таблицы стилей CSS и XML	10
Flash-ролики.....	11
FTP-клиенты	11
Удаленный доступ к серверу. Протокол SSH.....	12
ГЛАВА 2. БЫСТРЫЙ СТАРТ	13
2.1. Скрипты.....	13
2.2. Начальные и конечные теги	16
2.3. Использование точки с запятой	18
2.4. Составные выражения. Фигурные скобки	19
2.5. Комментарии.....	21

ГЛАВА 3. ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ	23
3.1. Объявление переменной. Оператор =	23
3.2. Типы данных.....	24
3.3. Целые числа	25
3.4. Вещественные числа	27
3.5. Строки	28
3.6. Кавычки	28
3.7. Оператор <<<.....	32
3.8. Обращение к неинициализированной переменной. Замечания (Notice)	32
3.9. Специальный тип <i>NULL</i>	34
3.10. Логический тип.....	35
3.11. Уничтожение переменной. Конструкция <i>unset()</i>	36
3.12. Проверка существования переменной. Конструкции <i>isset()</i> и <i>empty()</i>	36
3.13. Определение типа переменной	38
3.14. Неявное приведение типов	44
3.15. Явное приведение типов.....	46
3.16. Динамические переменные	51
ГЛАВА 4. КОНСТАНТЫ.....	53
4.1. Объявление константы. Функция <i>define()</i>	53
4.2. Функции для работы с константами.....	57
4.3. Динамически константы. Функция <i>constant()</i>	58
4.4. Проверка существования константы	59
4.5. Предопределенные константы	60
ГЛАВА 5. ОПЕРАТОРЫ И КОНСТРУКЦИИ ЯЗЫКА.....	63
5.1. Объединение строк. Оператор "точка"	63
5.2. Конструкция <i>echo</i> . Оператор "запятая"	64
5.3. Арифметические операторы.....	65
5.4. Поразрядные операторы	70
5.5. Операторы сравнения.....	75
5.6. Условный оператор <i>if</i>	79
5.7. Логические операторы.....	81
5.8. Условный оператор <i>x ? y : z</i>	89
5.9. Переключатель <i>switch</i>	90
5.10. Цикл <i>while</i>	95
5.11. Цикл <i>do ... while</i>	101

5.12. Цикл <i>for</i>	102
5.13. Включение файлов	107
5.14. Подавление вывода ошибок. Оператор @	113
5.15. Приоритет выполнения операторов.....	114
ГЛАВА 6. МАССИВЫ	117
6.1. Создание массива	117
6.2. Ассоциативные и индексные массивы	124
6.3. Многомерные массивы	129
6.4. Интерполяция элементов массива в строки.....	130
6.5. Конструкция <i>list()</i>	131
6.6. Обход массива	134
6.7. Цикл <i>foreach</i>	138
6.8. Проверка существования элементов массива.....	140
6.9. Количество элементов в массиве	144
6.10. Сумма элементов массива	146
6.11. Случайные элементы массива.....	147
6.12. Сортировка массивов	149
6.13. Суперглобальные массивы. Массив <code>\$_SERVER</code>	159
6.13.1. Элемент <code>\$_SERVER['DOCUMENT_ROOT']</code>	159
6.13.2. Элемент <code>\$_SERVER['HTTP_REFERER']</code>	160
6.13.3. Элемент <code>\$_SERVER['HTTP_USER_AGENT']</code>	161
6.13.4. Элемент <code>\$_SERVER['REMOTE_ADDR']</code>	161
6.13.5. Элемент <code>\$_SERVER['SCRIPT_FILENAME']</code>	162
6.13.6. Элемент <code>\$_SERVER['SERVER_NAME']</code>	162
6.13.7. Элемент <code>\$_SERVER['QUERY_STRING']</code>	163
6.13.8. Элемент <code>\$_SERVER['PHP_SELF']</code>	164
ГЛАВА 7. ФУНКЦИИ	165
7.1. Объявление и вызов функции	165
7.2. Параметры функции.....	168
7.3. Передача параметров по значению и ссылке.....	169
7.4. Необязательные параметры.....	170
7.5. Переменное количество параметров	172
7.6. Глобальные переменные.....	174
7.7. Статические переменные.....	175
7.8. Возврат массива функцией.....	176
7.9. Рекурсивные функции.....	177

7.10. Вложенные функции	179
7.11. Динамическое имя функции	179
7.12. Анонимные функции	180
7.13. Проверка существования функции	182
7.14. Неявное выполнение функций. Оператор <i>declare()</i>	188
7.15. Вспомогательные функции	193
ГЛАВА 8. ВЗАИМОДЕЙСТВИЕ PHP С HTML	197
8.1. Передача параметров методом GET	197
8.2. HTML-форма и ее обработчик	202
8.3. Текстовое поле	207
8.4. Поле для приема пароля	208
8.5. Текстовая область	209
8.6. Скрытое поле	210
8.7. Флажок	211
8.8. Список	213
8.9. Переключатель	215
8.10. Загрузка файла на сервер	217
ГЛАВА 9. СТРОКОВЫЕ ФУНКЦИИ	221
9.1. Функции для работы с символами	221
9.2. Поиск в строке	225
9.3. Замена в тексте	231
9.4. Преобразование регистра	237
9.5. Работа с HTML-кодом	238
9.6. Экранирование	247
9.7. Форматный вывод	250
9.8. Преобразование кодировок	256
9.9. Сравнение строк	259
9.10. Хранение данных	265
9.11. Работа с путями к файлам и каталогам	269
9.12. Объединение и разбиение строк	271
ГЛАВА 10. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ	283
10.1. Как изучать регулярные выражения?	283
10.2. Синтаксис регулярных выражений	284

10.3. Функции для работы с регулярными выражениями	288
10.4. Функции <i>preg_match()</i>	289
10.5. Функция <i>preg_match_all()</i>	294
10.6. Функция <i>preg_replace()</i>	297
10.7. Функция <i>preg_replace_callback()</i>	302
10.8. Функция <i>preg_split()</i>	304
10.9. Функция <i>preg_quote()</i>	306
ГЛАВА 11. ДАТА И ВРЕМЯ	309
11.1. Формирование даты и времени	309
11.2. Географическая привязка	316
11.3. Форматирование даты и времени	322
ГЛАВА 12. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ	337
12.1. Предопределенные константы	337
12.2. Поиск максимума и минимума	338
12.3. Генерация случайных чисел.....	340
12.4. Преобразование значений между различными системами счисления.....	342
12.5. Округление чисел	346
12.6. Логарифмические и степенные функции.....	349
12.7. Тригонометрические функции.....	353
12.8. Информационные функции	355
ГЛАВА 13. ФАЙЛЫ И КАТАЛОГИ	363
13.1. Создание файлов	363
13.2. Манипулирование файлами	370
13.3. Чтение и запись файлов	373
13.3.1. Чтение файлов.....	376
13.3.2. Запись файлов	383
13.3.3. Обязательно ли закрывать файлы?	387
13.3.4. Дозапись файлов.....	389
13.3.5. Блокировка файлов.....	390
13.3.6. Прямое манипулирование файловым указателем	395
13.4. Права доступа	399
13.5. Каталоги	403

ГЛАВА 14. HTTP-ЗАГОЛОВКИ	411
14.1. Функции для управления HTTP-заголовками	412
14.2. Кодировка страницы	414
14.3. HTTP-коды состояния	415
14.4. Список HTTP-заголовков	416
14.5. Подавление кэширования	419
ГЛАВА 15. COOKIE	425
ГЛАВА 16. СЕССИИ	431
ГЛАВА 17. ЭЛЕКТРОННАЯ ПОЧТА	437
17.1. Отправка почтового сообщения	437
17.2. Рассылка писем	439
ГЛАВА 18. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ВОЗМОЖНОСТИ PHP	441
18.1. Введение в объектно-ориентированное программирование	441
18.2. Создание класса	443
18.3. Создание объекта	443
18.4. Инкапсуляция. Спецификаторы доступа	445
18.5. Методы класса. Член <i>\$this</i>	447
18.6. Специальные методы класса	451
18.7. Функции для работы с методами и классами	452
18.8. Конструктор. Метод <code>__construct()</code>	454
18.9. Параметры конструктора	457
18.10. Деструктор. Метод <code>__destruct()</code>	459
18.11. Автозагрузка классов. Функция <code>__autoload()</code>	460
18.12. Аксессуары. Методы <code>__set()</code> и <code>__get()</code>	461
18.13. Проверка существования члена класса. Метод <code>__isset()</code>	463
18.14. Уничтожение члена класса. Метод <code>__unset()</code>	464
18.15. Динамические методы. Метод <code>__call()</code>	466
18.16. Интерполяция объекта. Метод <code>__toString()</code>	468
18.17. Наследование	470

18.18. Спецификаторы доступа и наследование	473
18.19. Перегрузка методов.....	476
18.20. Полиморфизм.....	478
18.21. Абстрактные классы	480
18.22. Абстрактные методы.....	481
18.23. Создание интерфейса	483
18.24. Реализация нескольких интерфейсов	485
18.25. Наследование интерфейсов	486
18.26. Статические члены класса.....	487
18.27. Статические методы класса.....	490
18.28. Константы класса	491
18.29. Предопределенные константы	493
18.30. <i>Final</i> -методы класса	494
18.31. <i>Final</i> -классы	496
18.32. Клонирование объекта	497
18.33. Управление процессом клонирования. Метод <i>__clone()</i>	498
18.34. Управление сериализацией. Методы <i>__sleep()</i> и <i>__wakeup()</i>	500
18.35. Синтаксис исключений	509
ГЛАВА 19. РАБОТА С СУБД MYSQL	513
19.1. Введение в СУБД и SQL.....	514
19.2. Первичные ключи.....	517
19.3. Создание и удаление базы данных	519
19.4. Выбор базы данных.....	521
19.5. Типы данных.....	523
19.6. Создание и удаление таблиц	529
19.7. Вставка числовых значений в таблицу.....	536
19.8. Вставка строковых значений в таблицу	538
19.9. Вставка календарных значений	540
19.10. Вставка уникальных значений	543
19.11. Механизм <i>AUTO_INCREMENT</i>	544
19.12. Многострочный оператор <i>INSERT</i>	544
19.13. Удаление данных.....	545
19.14. Обновление записей.....	547
19.15. Выборка данных	549
19.16. Условная выборка	551
19.17. Псевдонимы столбцов.....	558
19.18. Сортировка записей.....	558
19.19. Вывод записей в случайном порядке	561
19.20. Ограничение выборки	562

19.21. Вывод уникальных значений	563
19.22. Объединение таблиц	565
ГЛАВА 20. ВЗАИМОДЕЙСТВИЕ MYSQL И PHP	569
20.1. Функция <i>mysql_connect()</i>	569
20.2. Функция <i>mysql_close()</i>	571
20.3. Функция <i>mysql_select_db()</i>	572
20.4. Функция <i>mysql_query()</i>	573
20.5. Функция <i>mysql_result()</i>	575
20.6. Функция <i>mysql_fetch_row()</i>	576
20.7. Функция <i>mysql_fetch_assoc()</i>	577
20.8. Функция <i>mysql_fetch_array()</i>	580
20.9. Функция <i>mysql_fetch_object()</i>	582
20.10. Функция <i>mysql_num_rows()</i>	583
ЗАКЛЮЧЕНИЕ	587
Online-поддержка	588
Портал по программированию <i>SoftTime.ru</i>	588
Портал <i>Softtime.org</i>	590
Сайт <i>Softtime.biz</i>	590
ПРИЛОЖЕНИЯ	593
ПРИЛОЖЕНИЕ 1. УСТАНОВКА И НАСТРОЙКА PHP, WEB-СЕРВЕРА АПАЧЕ И MYSQL-СЕРВЕРА	595
П1.1. Где взять дистрибутивы?	595
П1.1.1. Дистрибутив PHP	596
П1.1.2. Дистрибутив Apache	597
П1.1.3. Дистрибутив MySQL	598
П1.2. Установка Web-сервера Apache под Windows.....	599
П1.3. Установка Web-сервера Apache под Linux	601
П1.4. Настройка виртуальных хостов.....	602
П1.5. Настройка кодировки по умолчанию	606
П1.6. Управление запуском и остановкой Web-сервера Apache	607

П1.7. Управление Apache из командной строки	608
П1.8. Установка PHP под Windows	609
П1.8.1. Установка PHP в качестве модуля	609
П1.8.2. Установка PHP как CGI-приложения.....	610
П1.9. Установка PHP под Linux	612
П1.10. Общая настройка конфигурационного файла php.ini	613
П1.11. Настройка и проверка работоспособности расширений PHP	616
ПРИЛОЖЕНИЕ 2. УСТАНОВКА MySQL	618
П2.1. Установка MySQL под Windows.....	618
П2.1.1. Процесс установки.....	618
П2.1.2. Постинсталляционная настройка	624
П2.1.3. Проверка работоспособности MySQL	631
П2.2. Установка MySQL под Linux	634
П2.3. Конфигурационный файл	637
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ.....	641



ГЛАВА 2

Быстрый старт

Несмотря на то, что PHP является универсальным языком программирования и может применяться для разработки практически любого программного обеспечения, основная его специализация — Web-разработка. Именно с этой целью он и проектировался, поэтому содержит множество инструментов для работы в Интернете.

ЗАМЕЧАНИЕ

В данной и последующих главах будем исходить из того, что интерпретатор PHP и Web-сервер установлены у вас на локальной машине и связаны таким образом, что скрипты, помещенные в файл с расширением `php`, обрабатываются интерпретатором, и по запросу к Web-серверу в браузере выводится результат работы. Если у вас не настроена связка Web-сервера и PHP, вы можете обратиться к приложениям 1 и 2, в которых детально описывается процесс развертывания Web-среды в операционных системах Windows и UNIX.

2.1. Скрипты

Язык программирования PHP считается скриптовым языком, поэтому программы, написанные на нем, называют *скриптами*. Главное отличие традиционных программ от скриптов заключается в том, что скрипты работают только в определенной среде и используют ресурсы данной среды. Программы, будь то компилируемые или интерпретируемые, не зависят от какого-то стороннего программного обеспечения. Например, скриптовый язык программирования JavaScript работает только в Web-браузерах, Visual Basic for Applications только в среде Microsoft Office. С использованием этих языков программирования невозможно создать программу, работающую без соответствующей среды. В случае PHP в качестве такой среды выступает Web-окружение (Web-сервер, сервер базы данных, почтовый сервер и т. п.).

ЗАМЕЧАНИЕ

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

Одной из главных особенностей языка программирования PHP является тот факт, что его код может располагаться вперемешку с HTML-кодом. Для того чтобы интерпретатор PHP различал HTML- и PHP-код, последний заключается в специальные теги `<?php` и `?>`, между которыми располагаются конструкции и операторы языка программирования PHP. В листинге 2.1 приводится классический пример, выводящий в окно браузера при помощи конструкции `echo` фразу "Hello world!" ("Привет мир!"). Содержимое листинга 2.1 следует поместить в файл с расширением `php`, например, в файл `index.php`. По умолчанию файлы с другими расширениями не обрабатываются PHP-интерпретатором, и PHP-код остается необработанным. Впрочем, такое поведение Web-сервера можно изменить (см. приложение 1).

ЗАМЕЧАНИЕ

`echo` — это конструкция языка программирования, позволяющая вывести одну или несколько строк в окно браузера. Более подробно конструкция `echo` обсуждается в разд. 5.2, посвященном строковым функциям.

Листинг 2.1. Простейший PHP-скрипт, `index.php`

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    <?php
      echo "Hello world!";
    ?>
  </body>
</html>
```

В результате работы скрипта из листинга 2.1 в окно браузера будет выведена фраза "Hello world!" (рис. 2.1).

При работе с серверными языками программирования, такими как PHP, следует помнить, что скрипты, расположенные между тегами `<?php` и `?>`, выполняются на сервере. Клиенту приходит лишь результат работы PHP-кода, в чем можно легко убедиться, просмотрев исходный код HTML-страницы. В листинге 2.2 приводится результат работы PHP-скрипта из листинга 2.1, который браузер получает от сервера.

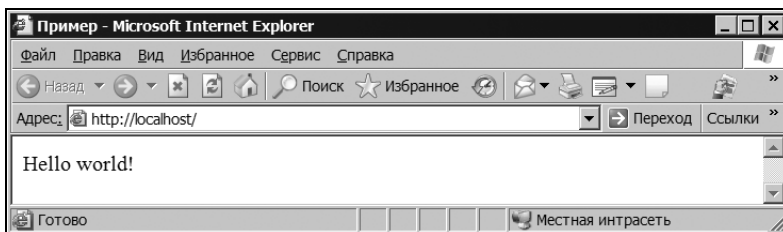


Рис. 2.1. Результат работы скрипта из листинга 2.1

ЗАМЕЧАНИЕ

Для просмотра исходного текста необходимо щелкнуть правой кнопкой мыши по странице и выбрать в контекстном меню соответствующий пункт, название которого зависит от браузера. В Internet Explorer необходимо выбрать пункт **Просмотр HTML-кода**, в Opera — **Исходный текст**, в FireFox — **View Page Source**.

Листинг 2.2. Исходный код HTML-страницы, которую получает браузер

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    Hello world! </body>
</html>
```

Если связка PHP и Web-сервера настроена неправильно, или PHP-скрипт был размещен в файле с расширением, отличным от php, например, в файле index.html, то в исходном коде HTML-страницы вместо содержимого листинга 2.2 можно увидеть содержимое листинга 2.1. В этом случае все, что расположено между тегами `<?php` и `?>`, будет рассматриваться браузером как один большой неизвестный ему HTML-тег, а окно браузера будет пустым (рис. 2.2).

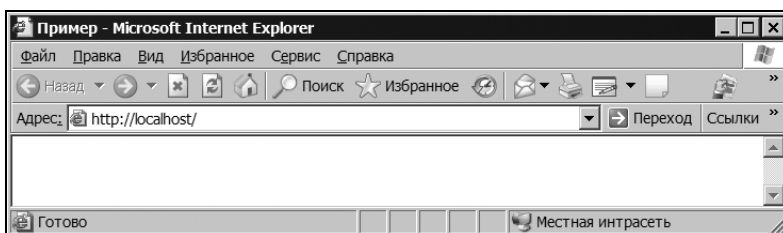


Рис. 2.2. PHP-код не подвергся интерпретации

2.2. Начальные и конечные теги

Как было указано в предыдущем разделе, PHP-скрипт должен быть размещен между начальным тегом `<?php` и конечным тегом `?>` для того, чтобы интерпретатор мог разделить HTML- и PHP-коды. Даже если HTML-код не используется, указание PHP-тегов является обязательным, в противном случае PHP-код будет выведен в окно браузера как есть, без интерпретации. Помимо тегов `<?php` и `?>`, PHP поддерживает еще ряд тегов, представленных в табл. 2.1.

ЗАМЕЧАНИЕ

До версии PHP 6.0 поддерживались начальные и конечные теги, характерные для ASP-скриптов, `<%` и `%>`, включить которые можно было в конфигурационном файле `php.ini`, назначив директиве `asp_tags` значение `On`. В версии PHP 6.0 данный тип тегов был исключен.

Таблица 2.1. Варианты PHP-тегов

Вариант тегов	Описание
<code><?php ... ?></code>	Классический вариант PHP-тегов, именно его рекомендуется использовать в повседневной практике
<code><? ... ?></code>	Краткий вариант PHP-тегов, работает только в том случае, если в конфигурационном файле <code>php.ini</code> включена (т. е. принимает значение <code>On</code>) директива <code>short_open_tag</code> . По умолчанию эта директива включена, однако все-таки лучше ориентироваться на полный вариант PHP-тегов <code><?php ... ?></code> . Последнее особенно актуально при использовании PHP совместно с XML-кодом, который использует теги <code><?xml ... ?></code> . При использовании коротких тегов может возникнуть неоднозначность в интерпретации
<code><script language="php"> ... </script></code>	Расширенный вариант PHP-тегов, так же как и вариант <code><?php ... ?></code> , доступен в любой момент без дополнительных настроек конфигурационного файла <code>php.ini</code> . Данный вариант несколько громоздок и распознается не всеми PHP-редакторами, тем не менее, следует быть готовыми встретить и такой вариант в PHP-скриптах
<code><?= ... ?></code>	Специальный вид тегов, предназначенный для вывода простого выражения в окно браузера. Конструкция <code><?= ... ?></code> эквивалентна <code><?php echo ... ?></code>

Отдельно следует отметить конструкцию `<?= ... ?>`, которую удобно использовать, когда скрипт состоит из одного выражения.

Например, скрипт из листинга 2.1 можно было бы переписать так, как это продемонстрировано в листинге 2.3.

Листинг 2.3. Использование тегов `<?= ... ?>`

```
<html>
  <head>
    <title>Пример</title>
  </head>
  <body>
    <?= "Hello world!" ?>
  </body>
</html>
```

Следует отметить, что HTML-страница может содержать более чем одну PHP-вставку. В листинге 2.4 приводится пример, который содержит две вставки: одна задает название страницы (в HTML-теге `<title>`), а вторая определяет содержимое страницы (в HTML-теге `<body>`).

ЗАМЕЧАНИЕ

В листинге 2.4 используется функция `date()`, которая возвращает текущую дату и время. Более подробно эта функция обсуждается в *главе 11*, посвященной функциям даты и времени.

Листинг 2.4. Допускается несколько PHP-вставок в HTML-код

```
<html>
  <head>
    <title><?php echo "Вывод текущей даты" ?></title>
  </head>
  <body>
    <?php
      echo "Текущая дата:<br>";
      echo date (DATE_RSS);
    ?>
  </body>
</html>
```

Результат работы скрипта из листинга 2.4 представлен на рис. 2.3.

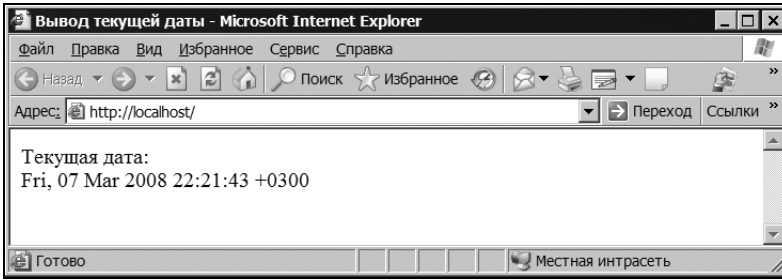


Рис. 2.3. Результат работы скрипта из листинга 2.4

2.3. Использование точки с запятой

Совокупность конструкций языка программирования, завершающуюся точкой с запятой, будем называть *выражением*.

Как видно из листинга 2.4, после строки "Вывод текущей даты" не указывается точка с запятой. Выражение одно, и надобность отделять его от других выражений отсутствует. Однако, как можно видеть во второй вставке, в конце каждой из конструкций `echo` имеется точка с запятой. Если забыть указать этот разделитель, интерпретатор языка программирования PHP посчитает выражение на новой строке продолжением предыдущего и не сможет корректно разобрать скрипт. В результате будет сгенерировано сообщение об ошибке "Parse error: syntax error, unexpected T_ECHO, expecting ',' or ';'". ("Ошибка разбора: синтаксическая ошибка, неожиданно встречена конструкция `echo`, ожидается либо запятая ',', либо точка с запятой ';'").

Последнее выражение перед завершающим тегом `?>` может не снабжаться точкой с запятой не только в тегах `<?= ... ?>`, но и во всех остальных вариантах, рассмотренных в табл. 2.1. Например, в листинге 2.4 после выражения `echo date(DATE_RSS)` точку с запятой можно не указывать. Однако настоятельно рекомендуется не пользоваться этой особенностью и помещать точки с запятой после каждого выражения, т. к. добавление новых операторов может привести к появлению трудноулавливаемых ошибок.

Переводы строк никак не влияют на интерпретацию скрипта, выражение может быть разбито на несколько строк — интерпретатор PHP будет считать, что выражение закончено лишь после того, как обнаружит точку с запятой или завершающий тег `?>`. В листингах 2.5 и 2.6 представлены два скрипта, аналогичные по своей функциональности.

Листинг 2.5. Использование точки с запятой

```
<?php
echo 5 + 5;
```

```
echo 5 - 2;  
echo "Hello world!";  
?>
```

Листинг 2.6. Альтернативная запись скрипта из листинга 2.5

```
<?php  
echo (5  
    +  
    5); echo (5-  
    2); echo("Hello world!"  
    );  
?>
```

ЗАМЕЧАНИЕ

Следует избегать конструкций, подобных той, которая приведена в листинге 2.6. Понятно написанный код — залог успеха, т. к. такой код намного проще и быстрее отлаживать. А искусство программиста, как известно, состоит не в безошибочном написании кода (ошибки делают все), а в умении его отлаживать.

2.4. Составные выражения. Фигурные скобки

Фигурные скобки позволяют объединить несколько выражений в группу, которую обычно называют *составным выражением* (листинг 2.7).

Листинг 2.7. Составное выражение

```
<?php  
{  
    echo 5 + 5;  
    echo 5 - 2;  
    echo "Hello world!";  
}  
?>
```

Само по себе составное выражение практически никогда не используется, основное его предназначение — работа совместно с условными операторами, операторами цикла и т. п., которые подробно рассматриваются в *главе 5*.

Составное выражение может быть расположено в нескольких РНР-вставках. В листинге 2.8 приводится пример двух составных выражений, которые разбиты несколькими РНР-вставками. Задача скрипта сводится к случайному выводу в окно браузера либо зеленого слова "Истина", либо красного слова "Ложь". Без использования фигурных скобок оператор `if` распространял бы свое действие только на одно выражение, использование составного выражения позволяет распространить его действие на несколько простых выражений.

ЗАМЕЧАНИЕ

Логический оператор `if` рассматривается в *главе 5*, а функция `rand()`, генерирующая случайное число из заданного интервала, обсуждается в *главе 12*.

Листинг 2.8. Составное выражение может располагаться в нескольких РНР-вставках

```
<?php
  if(rand(0,1))
  {
    ?>
    <div style='color:green'>
    <?php
      echo "Истина";
    ?>
    </div>
    <?php
  }
  else
  {
    ?>
    <div style='color:red'>
    <?php
      echo "Ложь";
    ?>
    </div>
    <?php
  }
?>
```

Как видно из листинга 2.8, составное выражение в любой момент может быть прервано тегами `<?php` и `?>`, а затем продолжено. Впрочем, существуют исключения, например, составное выражение, применяемое для формирования класса, нельзя

разбивать тегами `<?php` и `?>`. Возможно, в следующих версиях языка это ограничение будет устранено.

2.5. Комментарии

Код современных языков программирования является достаточно удобным для восприятия человеком по сравнению с машинными кодами, ассемблером или первыми языками программирования высокого уровня. Тем не менее, конструкции языка продиктованы архитектурой компьютера, и, создавая программы, разработчик волей не волей использует компьютерную, а не человеческую логику. Это, зачастую приводит к созданию достаточно сложных построений, которые нуждаются в объяснении на обычном языке. Для вставки таких пояснений в код предназначены *комментарии*.

PHP предоставляет несколько способов для вставки комментариев, варианты которых представлены в табл. 2.2.

Таблица 2.2. Варианты комментариев

Комментарий	Описание
<code>// ...</code>	Комментарий в стиле языка C++, начинающийся с символа двух слэшей (<code>//</code>) и заканчивающийся переводом строки
<code># ...</code>	Комментарий в стиле скриптовых языков UNIX, начинающийся с символа диеза (<code>#</code>) и заканчивающийся переводом строки
<code>/* ... */</code>	Если два предыдущих комментария ограничены лишь одной строкой, то комментарий в стиле языка C <code>/* ... */</code> является многострочным

В листинге 2.9 демонстрируется использование всех трех видов комментариев из табл. 2.2.

Листинг 2.9. Комментарии

```
<?php
echo "Hello"; // это комментарий
echo "Hello"; # и это комментарий
/*
  а это многострочный
  комментарий
*/
?>
```

Естественно, что комментарии PHP действуют только внутри тегов-ограничителей `<?php ... ?>`. То есть если символы комментариев будут находиться вне тегов-ограничителей, то они, как и любой текст, будут отображены браузером (листинг 2.10).

Листинг 2.10. Комментарии действуют только внутри тегов `<?php` и `?>`

```
<?php
    echo "Hello<br>"; // нормальный комментарий
?>
// этот текст отобразится браузером.
<!-- Этот текст не будет отображен браузером, поскольку заключен между симво-
лами, являющимися комментариями HTML. Однако он может быть просмотрен в исход-
ном коде HTML-страницы -->
```

Результат работы скрипта из листинга 2.10 представлен на рис. 2.4.

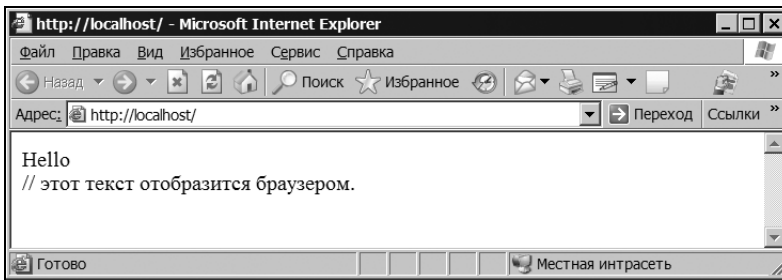


Рис. 2.4. Результат работы скрипта из листинга 2.10

Комментарии можно вставлять не только после выражения после точки с запятой, но и посередине. Так, скрипт, представленный в листинге 2.11, является вполне корректным.

Листинг 2.11. Комментарий в списке аргументов функции

```
<?php
    echo strstr( // эту функцию мы рассмотрим позднее
                "Hello, world", "H");
?>
```



ГЛАВА 3

Переменные и типы данных

Ключевым объектом практически любого языка программирования является переменная. Под *переменной* в общем случае понимается именованная область памяти. В этой области может храниться либо строка, либо число, которыми можно манипулировать при помощи имени переменной (его мы далее для простоты будем называть просто переменной). То, что хранится в области памяти, будем называть *значением переменной*.

3.1. Объявление переменной.

Оператор =

В PHP переменные начинаются со знака доллара (\$), за которым может следовать любое количество буквенно-цифровых символов и символов подчеркивания, но первый символ не может быть цифрой. Таким образом, допустимы следующие имена переменных: \$n, \$n1, \$user_func_5 и т. д. В отличие от ключевых слов, имена переменных в PHP *чувствительны к регистру*, т. е. переменные \$user, \$User и \$USER являются различными (листинг 3.1).

Листинг 3.1. Зависимость переменных от регистра

```
<?php
    $user = "Владимир";
    $User = "Дмитрий";
    $USER = "Юрий";
    echo $user; // Владимир
    echo $User; // Дмитрий
    echo $USER; // Юрий
?>
```

Как видно из листинга 3.1, для присвоения значения переменной необходимо воспользоваться оператором присваивания `=`, который позволяет инициализировать переменную.

При объявлении числовых значений в качестве разделителя целого значения и дробной части выступает точка (листинг 3.2).

Листинг 3.2. Объявление чисел

```
<?php
    $number = 1;
    $var = 3.14;
?>
```

Допускается инициализация одним значением сразу нескольких переменных за счет того, что оператор `=` возвращает результат присвоения. В листинге 3.3 переменным `$num`, `$number` и `$var` присваивается значение 1 в одну строку за счет использования оператора `=` в цепочке.

Листинг 3.3. Инициализация трех переменных одним значением

```
<?php
    $num = $number = $var = 1;
?>
```

3.2. Типы данных

Язык PHP является слаботипизированным, т. е. переменные языка не требуют строгого задания типа при их объявлении, а в ходе выполнения программы тип переменной может быть практически всегда изменен неявным образом без специальных преобразований. Например, переменная, объявленная строкой, может использоваться далее в арифметических операциях, выступать как логическая переменная, а в конце ей в качестве значения может быть присвоен объект. Все это позволяет разработчику практически не задумываться о типах данных. Тем не менее, некоторые ограничения на типы переменных все-таки накладываются, поэтому о типах переменных все же следует иметь представление. В табл. 3.1 представлены типы данных, которые поддерживаются PHP.

ЗАМЕЧАНИЕ

Тип `int64` планируется ввести в PHP 6.0.

Таблица 3.1. Типы данных PHP

Тип данных	Описание
boolean	Логический тип, способный принимать лишь два значения: TRUE (истина) и FALSE (ложь). Более подробно данный тип обсуждается в <i>главе 5</i>
integer	Целое число, разрядность которого зависит от архитектуры компьютера. Если разрядность операционной системы составляет 32 бита, то число может принимать значения от $-2\,147\,483\,648$ до $2\,147\,483\,647$. Если разрядность составляет 64 бита, то число может принимать значение от $-9\,223\,372\,036\,854\,775\,808$ до $9\,223\,372\,036\,854\,775\,807$
int64	Целое число, разрядность которого не зависит от архитектуры компьютера и всегда равна 64 битам. Таким образом, переменные данного типа могут принимать значения от $-9\,223\,372\,036\,854\,775\,808$ до $9\,223\,372\,036\,854\,775\,807$
double (или float)	Вещественное число, минимально возможное значение которого составляет $\pm 1.7 \times 10^{-308}$, а максимально возможное $\pm 1.7 \times 10^{308}$
string	Строковый тип, может хранить строку произвольного объема
array	Массив — это объединение нескольких однотипных переменных под одним именем. Обращаться к отдельным переменным можно при помощи индекса массива. Более подробно массивы обсуждаются в <i>главе 6</i>
object	Объект. Это конструкция, объединяющая несколько разнотипных переменных и методы их обработки. Более подробно объекты обсуждаются в <i>главе 18</i>
resource	Дескриптор, позволяющий оперировать тем или иным ресурсом, доступ к которому осуществляется при помощи библиотечных функций. Дескрипторы применяются при работе с файлами, базами данных, динамическими изображениями и т. д. Более подробно дескрипторы будут рассмотрены в соответствующих главах
NULL	Специальный тип, который сигнализирует о том, что переменная не была инициализирована

3.3. Целые числа

Целые числа являются наиболее распространенными в программировании. Это связано с тем, что большинство прикладных задач носит арифметический характер,

а также с тем, что это наиболее быстродействующий тип данных. Если для обработки чисел с плавающей точкой используется математический сопроцессор с достаточно сложными алгоритмами их обработки, то оперировать целыми числами процессор может напрямую.

ЗАМЕЧАНИЕ

В отличие от других языков программирования переполнения (т. е. выхода значения за допустимые границы) в PHP не бывает, если полученное значение не убирается в `integer`, для него автоматически выбирается больший тип данных (`int64` или `double`).

Язык PHP является C-подобным, поэтому целочисленная переменная может быть объявлена несколькими способами (листинг 3.4).

Листинг 3.4. Объявление целочисленных переменных

```
<?php
$num = 1234; // десятичное число
$num = +123; // десятичное число
$num = -123; // отрицательное число
$num = 0123; // восьмеричное число (эквивалентно 83)
$num = 0x1A; // шестнадцатеричное число (эквивалентно 26)
?>
```

Целое положительное число объявляется, как правило, без указания плюса перед ним (впрочем, указанием символа плюса + не приводит к ошибке). Для объявления отрицательного числа перед ним размещается символ минуса -.

По умолчанию считается, что числа задаются в десятичной системе исчисления, однако PHP позволяет объявлять переменные в восьмеричной и шестнадцатеричной системах счисления. В восьмеричной системе счисления основанием служит число 8, а для выражения всех чисел используются цифры от 0 до 7. Число 8 в восьмеричной системе счисления имеет то же значение, что число 10 в десятичной. Шестнадцатеричная система счисления (основанием служит число 16) использует цифры от 0 до 9 и буквы английского алфавита от A до F, означающие шестнадцатеричные "цифры" 10, 11, 12, 13, 14 и 15. Числа, объявленные в восьмеричной системе счисления, предваряются префиксом 0, в шестнадцатеричной системе — префиксом 0x.

ЗАМЕЧАНИЕ

Может показаться, что восьмеричные и шестнадцатеричные числа являются избыточным наследием языка C, однако, как будет показано дальше, они применяются и в Web-разработке, например, при задании прав доступа к файлам и папкам.