

МАКСИМ КУЗНЕЦОВ
ИГОРЬ СИМДЯНОВ



РНР

практика создания Web-сайтов

2-е издание



ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ВОЗМОЖНОСТИ РНР

РАБОТА С БАЗАМИ ДАННЫХ

«ХИТРОСТИ» РНР

ТОНКОСТИ РАБОТЫ С HTTP-ПРОТОКОЛОМ

СИСТЕМА УПРАВЛЕНИЯ СОДЕРЖИМЫМ САЙТА (CMS)

РАЗРАБОТКА ДИНАМИЧЕСКИХ WEB-ПРИЛОЖЕНИЙ

СОЗДАНИЕ ДИНАМИЧЕСКИХ ИЗОБРАЖЕНИЙ

PRO

ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

**Максим Кузнецов
Игорь Симдянов**

РНР

**практика создания
Web-сайтов**

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2008

УДК 681.3.06
ББК 32.973.26-018.2
К89

Кузнецов, М. В.

К89 РНР. Практика создания Web-сайтов / М. В. Кузнецов, И. В. Симдянов. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 1264 с.: ил.
ISBN 978-5-9775-0203-0

Рассматривается создание большого количества Web-приложений, входящих в состав полнофункционального Web-сайта. Попутно подробно обсуждаются все вопросы, с которыми может столкнуться Web-разработчик, начиная с создания инструментария для быстрой разработки Web-приложений и последних нововведений языка программирования РНР и заканчивая вопросами безопасности и особенностями программирования клиент-серверных приложений.

Книга ориентирована на читателей, знакомых с языком разметки HTML и базовыми возможностями языка программирования РНР. Второе издание полностью переработано, учтены нововведения версий РНР 5.1 и 6.0.

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>Ирина Артемьева</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Иины Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.03.08.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 101,91.

Тираж 3000 экз. Заказ №

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

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

Оглавление

ВВЕДЕНИЕ.....	1
Для кого и о чем эта книга?.....	1
Как построена книга.....	2
Предисловие авторов ко второму изданию.....	4
Благодарности.....	4
ЧАСТЬ I. ОБЩИЕ ВОПРОСЫ.....	5
ГЛАВА 1. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ВОЗМОЖНОСТИ PHP.....	7
1.1. Введение в объектно-ориентированное программирование.....	7
1.2. Создание класса.....	9
1.3. Создание объекта.....	10
1.4. Инкапсуляция. Спецификаторы доступа.....	12
1.5. Методы класса. Член <i>\$this</i>	14
1.6. Дамп объекта.....	19
1.7. Специальные методы класса.....	20
1.8. Функции для работы с методами и классами.....	21
1.9. Конструктор. Метод <code>__construct()</code>	23
1.10. Параметры конструктора.....	26
1.11. Деструктор. Метод <code>__destruct()</code>	28
1.12. Автозагрузка классов. Функция <code>__autoload()</code>	29
1.13. Аксессоры. Методы <code>__set()</code> и <code>__get()</code>	30
1.14. Проверка существования члена класса. Метод <code>__isset()</code>	32
1.15. Уничтожение члена класса. Метод <code>__unset()</code>	33
1.16. Динамические методы. Метод <code>__call()</code>	35
1.17. Интерполяция объекта. Метод <code>__toString()</code>	38
1.18. Экспорт объектов. Метод <code>__set_state()</code>	40
1.19. Наследование.....	46

1.20. Спецификаторы доступа и наследование	49
1.21. Перегрузка методов	52
1.22. Полиморфизм	54
1.23. Абстрактные классы	57
1.24. Абстрактные методы	58
1.25. Создание интерфейса	59
1.26. Реализация нескольких интерфейсов	62
1.27. Наследование интерфейсов	63
1.28. Статические члены класса	64
1.29. Статические методы класса	68
1.30. Константы класса	69
1.31. Предопределенные константы	70
1.32. <i>Final</i> -методы класса	73
1.33. <i>Final</i> -классы	75
1.34. Клонирование объекта	76
1.35. Управление процессом клонирования. Метод <i>__clone()</i>	78
1.36. Сериализация объектов	79
1.37. Управление сериализацией. Методы <i>__sleep()</i> и <i>__wakeup()</i>	82
1.38. Синтаксис исключений	91
1.39. Интерфейс класса <i>Exception</i>	95

ГЛАВА 2. РАБОТА С СУБД MYSQL.....99

2.1. Введение в СУБД и SQL	100
2.2. Первичные ключи	104
2.3. Создание и удаление базы данных	105
2.4. Выбор базы данных	108
2.5. Типы данных	110
2.6. Создание и удаление таблиц	116
2.7. Вставка числовых значений в таблицу	124
2.8. Вставка строковых значений в таблицу	126
2.9. Вставка календарных значений	128
2.10. Вставка уникальных значений	131
2.11. Механизм <i>AUTO_INCREMENT</i>	132
2.12. Многострочный оператор <i>INSERT</i>	133
2.13. Удаление данных	134
2.14. Обновление записей	135
2.15. Выборка данных	138
2.16. Условная выборка	140
2.17. Псевдонимы столбцов	147

2.18. Сортировка записей.....	148
2.19. Вывод записей в случайном порядке.....	151
2.20. Ограничение выборки	151
2.21. Вывод уникальных значений.....	153
2.22. Объединение таблиц.....	155
2.23. Функции MySQL.....	157
2.24. PHP и MySQL.....	204
ГЛАВА 3. ПРОТОКОЛ HTTP.....	214
3.1. Функции для работы с <i>HTTP</i> -заголовками	215
3.2. Сессии и <i>cookie</i>	239
3.3. Сокеты и CURL.....	248
3.4. Работа с доменами и IP-адресами	282
ГЛАВА 4. "ХИТРОСТИ" PHP.....	289
4.1. PHP и JavaScript	289
4.2. О профилировании кода.....	291
4.3. Подсветка кода с помощью стандартных функций PHP	294
4.4. Подсветка синтаксиса PHP (собственная функция).....	295
4.5. Загрузка файлов на сервер	300
4.6. Редактирование файлов на сервере.....	304
4.7. Счетчик количества загрузок файла	307
4.8. Количество файлов в каталогах	310
4.9. Копирование содержимого одной директории в другую	313
4.10. Удаление директории	315
4.11. Случайное изображение из директории	316
4.12. Определение размера файла	317
4.13. Предотвращение загрузки страниц	319
ГЛАВА 5. БЕЗОПАСНОСТЬ СОЗДАВАЕМЫХ WEB-ПРИЛОЖЕНИЙ.....	324
5.1. Проверка корректности данных, вводимых пользователем.....	324
5.2. Публикация изображений и файлов	332
5.3. Методы шифрования	337
5.4. SQL-инъекции	345
5.5. XSS-инъекции	359

ГЛАВА 6. ВСПОМОГАТЕЛЬНЫЙ НАБОР КЛАССОВ. <i>FRAMEWORK</i>	366
6.1. Требования к набору классов	369
6.2. HTML-форма и ее обработчик.....	372
6.3. Обработка исключительных ситуаций	379
6.4. Базовый класс <i>field</i>	382
6.5. Текстовое поле. Класс <i>field_text</i>	386
6.6. Класс <i>from</i>	392
6.7. Пример HTML-формы.....	397
6.8. Поле для пароля. Класс <i>field_password</i>	407
6.9. Поле для ввода английского текста. Класс <i>field_text_english</i>	410
6.10. Поле для ввода целых чисел. Класс <i>field_text_int</i>	412
6.11. Поле для ввода электронной почты. Класс <i>field_text_email</i>	415
6.12. Текстовая область. Класс <i>field_textarea</i>	417
6.13. Скрытое поле. Класс <i>field_hidden</i>	427
6.14. Скрытое поле для целых значений. Класс <i>field_hidden_int</i>	431
6.15. Флажок. Класс <i>field_checkbox</i>	439
6.16. Список. Класс <i>field_select</i>	443
6.17. Переключатели. Класс <i>field_radio</i>	449
6.18. Поле для загрузки файла на сервер. Класс <i>field_file</i>	454
6.19. Заголовок. Класс <i>field_title</i>	460
6.20. Параграф. Класс <i>field_paragraph</i>	465
6.21. Выбор даты и времени. Класс <i>field_datetime</i>	468
6.22. Обзор элементов управления.....	474
ГЛАВА 7. ПОСТРАНИЧНАЯ НАВИГАЦИЯ	476
7.1. Базовый класс постраничной навигации	476
7.2. Файловая постраничная навигация	482
7.3. Постраничная навигация и поиск.....	488
7.4. Постраничная навигация для директории	493
7.5. Постраничная навигация для базы данных.....	499
7.6. Изменение формата постраничной навигации	507
ЧАСТЬ II. СОЗДАНИЕ САЙТА	511
ГЛАВА 8. ПРОЕКТИРОВАНИЕ САЙТА	513
8.1. Структура системы управления сайтом (CMS).....	515
8.2. Общие файлы системы администрирования.....	521

ГЛАВА 9. ОГРАНИЧЕНИЕ ДОСТУПА К СИСТЕМЕ АДМИНИСТРИРОВАНИЯ	529
ГЛАВА 10. НОВОСТНОЙ БЛОК	547
10.1. База данных	547
10.2. Система администрирования	548
10.3. Система представления	571
ГЛАВА 11. БЛОК "ВОПРОСЫ И ОТВЕТЫ"	581
11.1. База данных	581
11.2. Система администрирования	582
11.3. Система представления	608
ГЛАВА 12. СИСТЕМА АДМИНИСТРИРОВАНИЯ СОДЕРЖИМОГО САЙТА (CMS)	611
12.1. База данных	611
12.2. Система администрирования	621
12.3. Система представления	662
ГЛАВА 13. КАТАЛОГ ПРОДУКЦИИ (УСЛУГ)	678
13.1. Проектирование базы данных	678
13.2. Система администрирования	682
13.3. Импорт прайс-листа	712
13.4. Блок представления	722
ГЛАВА 14. СИСТЕМА ПОИСКА ПО САЙТУ	734
14.1. Специализированный поиск по каталогу	734
14.2. Поиск по сайту	747
ГЛАВА 15. БЛОК "КОНТАКТЫ"	762
15.1. База данных	762
15.2. Система администрирования	763
15.3. Блок представления	766

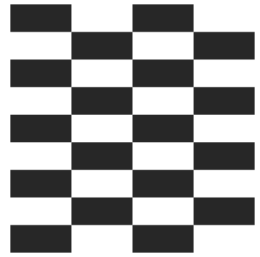
ГЛАВА 16. БЛОК ГОЛОСОВАНИЯ.....	769
16.1. База данных.....	769
16.2. Система администрирования.....	772
16.3. Система представления.....	784
ГЛАВА 17. ГОСТЕВАЯ КНИГА.....	793
17.1. База данных.....	793
17.2. Блок представления.....	795
17.3. Система администрирования.....	805
ГЛАВА 18. РЕГИСТРАЦИЯ ПОЛЬЗОВАТЕЛЕЙ.....	818
18.1. База данных.....	818
18.2. Регистрация пользователей.....	820
18.3. Аутентификация пользователя.....	826
18.4. Восстановление пароля.....	834
18.5. Система администрирования.....	838
ГЛАВА 19. ПОЧТОВАЯ РАССЫЛКА.....	846
ГЛАВА 20. ФОТОГАЛЕРЕЯ.....	853
20.1. База данных.....	853
20.2. Система администрирования.....	856
20.3. Система представления.....	882
ГЛАВА 21. FTP-МЕНЕДЖЕР.....	893
21.1. Функции для работы с FTP-сервером.....	894
21.2. FTP-менеджер.....	904
ГЛАВА 22. ЗАЩИТА ДИРЕКТОРИЙ ПАРОЛЕМ.....	935
22.1. Конфигурационные файлы .htaccess и .htpasswd.....	935
22.2. Web-интерфейс защиты директории паролем.....	943

ГЛАВА 23. СИСТЕМА МОНИТОРИНГА ПОЗИЦИЙ САЙТА В ПОИСКОВЫХ СИСТЕМАХ	967
23.1. Извлечение ссылок с Yandex	968
23.2. Извлечение ссылок с Google	972
23.3. Извлечение ссылок с Rambler	974
23.4. Извлечение ссылок с Aport	976
23.5. Мониторинг позиции сайта	978
ГЛАВА 24. СИСТЕМА УЧЕТА ПОСЕЩАЕМОСТИ САЙТА	987
24.1. База данных	988
24.2. Учет статистики	999
24.3. Система администрирования	1007
24.4. Разработка системы администрирования	1012
ГЛАВА 25. ФОРУМ: ПРОЕКТИРОВАНИЕ	1038
25.1. Проектирование базы данных	1039
25.2. Проектирование структуры	1052
ГЛАВА 26. ФОРУМ: СИСТЕМА ПРЕДСТАВЛЕНИЯ	1054
26.1. Описание файлов форума	1054
26.2. Описание функциональности форума	1058
ГЛАВА 27. ФОРУМ: СИСТЕМА АДМИНИСТРИРОВАНИЯ	1067
27.1. Описание файлов форума	1067
27.2. Описание функциональности форума	1069
ГЛАВА 28. ДИНАМИЧЕСКИЕ ИЗОБРАЖЕНИЯ. БИБЛИОТЕКА GDLib.....	1080
28.1. Информационные функции	1081
28.2. Функции создания изображений	1089
28.3. Функции сохранения и вывода изображений	1092
28.4. Функции преобразования изображений	1094

28.5. Функции для работы с цветом.....	1099
28.6. Функции рисования	1108
28.7. Функции настройки рисования	1119
28.8. Функции для работы с текстом	1122
ЗАКЛЮЧЕНИЕ.....	1130
ПРИЛОЖЕНИЯ	1131
ПРИЛОЖЕНИЕ 1. УСТАНОВКА И НАСТРОЙКА PHP, WEB-СЕРВЕРА APACHE И MYSQL-СЕРВЕРА	1133
П1.1. Где взять дистрибутивы?	1134
П1.2. Установка Web-сервера Apache под Windows	1137
П1.3. Установка Web-сервера Apache под Linux.....	1141
П1.4. Настройка виртуальных хостов.....	1142
П1.5. Настройка кодировки по умолчанию.....	1146
П1.6. Управление запуском и остановкой Web-сервера Apache.....	1147
П1.7. Управление Apache из командной строки.....	1148
П1.8. Установка PHP под Windows.....	1150
П1.9. Установка PHP под Linux	1153
П1.10. Общая настройка конфигурационного файла php.ini.....	1154
П1.11. Настройка и проверка работоспособности расширений PHP	1158
ПРИЛОЖЕНИЕ 2. УСТАНОВКА MYSQL	1160
П2.1. Установка MySQL под Windows	1160
П2.2. Установка MySQL под Linux	1178
П2.3. Конфигурационный файл.....	1182
П2.4. Утилита mysql.....	1185
П2.5. Перенос баз данных с одного сервера на другой.....	1197
ПРИЛОЖЕНИЕ 3. ИСПОЛЬЗОВАНИЕ CRON.....	1201
П3.1. PHP как консольный интерпретатор.....	1201
П3.2. Планировщик заданий или работа с cron	1205

ПРИЛОЖЕНИЕ 4. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ	1209
П4.1. Синтаксис регулярных выражений	1209
П4.2. Функции для работы с регулярными выражениями.....	1213
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА.....	1221
<i>HTML, XML, CSS, JavaScript и Flash</i>	1223
<i>PHP и Perl</i>	1226
СУБД <i>MySQL</i>	1228
Интернет и Web-сервер <i>Apache</i>	1230
Регулярные выражения	1231
<i>UNIX</i> -подобные операционные системы.....	1231
Методология программирования	1233
ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ.....	1235

ГЛАВА 1



Объектно-ориентированные возможности PHP

Последние два десятилетия в IT-индустрии получил широкое распространение объектно-ориентированный подход. Его введение связано со все возрастающим объемом программных систем, с которыми приходится сталкиваться разработчикам.

ЗАМЕЧАНИЕ

Данная глава кратко рассматривает объектно-ориентированные возможности PHP. Всестороннему рассмотрению предмета посвящена наша отдельная книга "Объектно-ориентированное программирование на PHP".

1.1. Введение в объектно-ориентированное программирование

Независимо от языка программирования объектно-ориентированный подход имеет ряд общих принципов, а именно:

- возможность создавать *абстрактные типы данных*, позволяющая наряду с predefined типами данных (такими как `integer`, `bool`, `double`, `string`) вводить свои собственные типы данных (классы) и объявлять "переменные" таких типов данных (объекты). Создавая свои собственные типы данных, программист оперирует не машинными терминами (переменная, функция), а объектами реального мира, поднимаясь тем самым на новый абстрактный уровень. Яблоки и людей нельзя умножать друг на

друга, однако низкоуровневый код запросто позволит совершить такую логическую ошибку, тогда как при использовании абстрактных типов данных такая операция становится невозможной;

- *инкапсуляция*, допускающая взаимодействие пользователя с абстрактными типами данных только через их интерфейс и скрывающая внутреннюю реализацию объекта, не допуская влияния на его внутреннее состояние. Память человека ограничена и не может содержать все детали огромного проекта, тогда как использование инкапсуляции позволяет разработать объект и использовать его, не заботясь о внутренней реализации, прибегая только к небольшому числу интерфейсных методов;
- *наследование*, позволяющее развить существующий абстрактный тип данных — класс, создав на его основе новый класс. При этом новый класс автоматически получает возможности уже существующего абстрактного типа данных. Зачастую абстрактные типы данных слишком сложны, поэтому прибегают к их последовательной разработке, выстраивая иерархию классов от общего к частному;
- *полиморфизм*, допускающий построение целых цепочек и разветвленных деревьев наследующих друг другу абстрактных типов данных (классов). При этом весь набор классов будет иметь ряд методов с одинаковыми названиями: любой из классов данного дерева гарантированно обладает методом с таким именем. Этот принцип помогает автоматически обрабатывать массивы данных разного типа.

Абстрактные типы данных необходимы для того, чтобы дать программисту вводить в программу переменные с желаемыми свойствами, так как возможностей существующих в языке типов данных зачастую не хватает. Связи между объектами реального мира зачастую настолько сложны, что для их эффективного моделирования необходим отдельный язык программирования. Разрабатывать специализированный язык программирования для каждой прикладной задачи — очень дорогое удовольствие. Поэтому в языки программирования вводится объектно-ориентированный подход, который позволяет создавать свой мини-язык путем создания классов и их объектов. Переменными такого мини-языка программирования являются программные *объекты*, в качестве типа для которых выступает класс. *Класс* описывает состав объекта — переменные и функции, которые обрабатывают переменные и тем самым определяют поведение объекта (рис. 1.1).

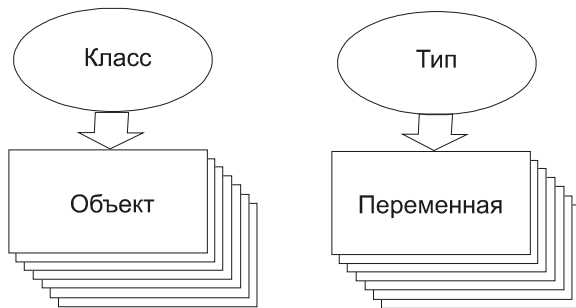


Рис. 1.1. Переменные объявляются при помощи типа, объекты при помощи класса

1.2. Создание класса

Класс объявляется при помощи ключевого слова `class`, после которого следует уникальное имя класса и тело класса в фигурных скобках. В теле класса объявляются переменные и функции класса, которые соответственно называются *методами* и *членами*. В листинге 1.1 приводится общий синтаксис объявления класса.

Листинг 1.1. Объявление класса

```
<?php
class имя_класса
{
    // Члены и
    // методы класса
}
?>
```

Важной особенностью PHP является то, что PHP-скрипты могут включаться в документ при помощи тегов `<?php` и `?>`. Один документ может содержать множество включений этих тегов, однако класс должен объявляться в одном неразрывном блоке `<?php` и `?>`. Попытка разорвать объявление класса приводит к генерации интерпретатором ошибки разбора **Parse error: parse error, unexpected ';', expecting T_FUNCTION**.

Так как прерывать объявление класса недопустимо, его не удастся механически разбить и при помощи инструкций `include()`, `include_once()`, `require()`, `require_once()`. Допускается, однако, использование этих конструкций внутри методов.

ЗАМЕЧАНИЕ

Напомним, что при помощи инструкций `include()`, `include_once()`, `require()`, `require_once()` можно включать в состав PHP-скриптов другие PHP-скрипты. Это позволяет разбивать объемные многострочные файлы на множество мелких файлов, которые программисту проще воспринять. При отсутствии включаемого файла инструкция `include()` генерирует предупреждение, однако не останавливает работу скрипта, в то время как `require()` в этом случае аварийно завершает работу приложения. Допускается множественное включение файлов друг в друга, что может приводить к запутанным ситуациям и многократному включению файлов в приложение. Суффикс `once` означает, что файл будет включен лишь один раз, и повторный вызов инструкции `include_once()` или `require_once()` игнорируется. Это особенно удобно для включения библиотек функций и классов, повторное объявление которых вызывает ошибку.

1.3. Создание объекта

Объект объявляется при помощи ключевого слова `new`, за которым следует имя класса. В листинге 1.2 создается пустой класс `emp` и объявляется объект `$obj` данного класса.

ЗАМЕЧАНИЕ

После имени класса могут следовать необязательные круглые скобки.

Листинг 1.2. Создание класса `emp` и объявление объекта `$obj` данного класса

```
<?php
class emp {}
$obj = new emp;
?>
```


ЗАМЕЧАНИЕ

Имена классов, в отличие от имен переменных и объектов, не зависят от регистра, поэтому можно использовать для объявления классов имена `emp`, `Emp()`, `EMP()`. Однако использование вместо объекта `$obj` переменной `$Obj` приведет к ошибке — интерпретатор PHP будет считать, что в программу введена новая переменная.

Объект `$obj` является обычной переменной PHP, правда, со специфичными свойствами. Как и любую другую переменную, объект можно передавать в качестве параметра функции, использовать как элемент массива или присваивать ему новое значение. В листинге 1.3 приводится пример, в котором объекту `$obj` по мере выполнения программы присваивается некоторое числовое значение, и `$obj` становится переменной числового типа.

Листинг 1.3. Объект — это обычная переменная

```
<?php
class emp {}
$obj = new emp();
$obj = 3;
echo $obj; // 3
?>
```

Объект существует до конца времени выполнения скрипта или пока не будет уничтожен явно при помощи конструкции `unset()` (листинг 1.4). Использование конструкции `unset()` может быть полезным, если объект занимает большой объем оперативной памяти и ее следует освободить, чтобы не допустить переполнения.

Листинг 1.4. Объект — это обычная переменная

```
<?php
// Объявление класса
class emp {}
// Создание объекта класса
$obj = new emp();
// Уничтожение объекта
unset($obj);
?>
```

В отличие от других языков программирования объект, объявленный внутри блока, ограниченного фигурными скобками, существует и вне его, не подвергаясь уничтожению при выходе из блока.

1.4. Инкапсуляция. Спецификаторы доступа

Как было показано в предыдущем разделе, класс может использоваться без методов и членов, однако пользы в этом случае от него не больше, чем от переменной, которая не может хранить значения. Как правило, классы содержат члены и методы. Часть из них будет использоваться внешним разработчиком, часть предназначена только для внутреннего использования в рамках класса. PHP предоставляет специальные ключевые слова — *спецификаторы доступа*, позволяющие указать, какие члены и методы доступны извне, а какие нет. Это позволяет реализовать принцип инкапсуляции.

Открытые члены класса объявляются спецификатором доступа `public` и доступны как методам класса, так и внешнему по отношению к классу коду. *Закрытые* методы и члены класса объявляются при помощи спецификатора `private` и доступны только в рамках класса; обратиться к ним извне невозможно.

ЗАМЕЧАНИЕ

Помимо спецификаторов `public` и `private` в PHP поддерживается спецификатор `protected`, используемый при наследовании и подробно рассматриваемый далее. В более ранних версиях для объявления члена класса использовалось ключевое слово `var`. Члены, объявленные с его помощью, были открытыми. В текущих версиях PHP по-прежнему допускается использовать ключевое слово `var` для объявления членов класса, однако это не рекомендуется, а само ключевое слово признано устаревшим. С большой долей вероятности оно будет исключено из PHP 6.

В листинге 1.5 представлен класс "сотрудник" (`employee`), который содержит четыре члена:

- `$surname` — фамилия сотрудника;
- `$name` — имя сотрудника;
- `$patronymic` — отчество сотрудника;
- `$age` — возраст сотрудника.

Листинг 1.5. Класс employee

```
<?php
class employee
{
    public $surname;
    public $name;
    public $patronymic;
    private $age;
}
?>
```

Рассмотрим пример. Пусть класс `employee` располагается в файле `class.employee.php`. В листинге 1.6 объявляется объект `$emp` класса `employee`, при этом члены класса получают необходимые значения и выводятся в окно браузера.

Листинг 1.6. Обращение к членам класса employee

```
<?php
// Подключаем объявление класса
require_once("class.employee.php");

// Объявляем объект класса employee
$emp = new employee();

// Присваиваем значения членам класса
$emp->surname    = "Борисов";
$emp->name       = "Игорь";
$emp->patronymic = "Иванович";
// $emp->age = 23; // Ошибка

// Выводим члены класса
echo $emp->surname." ". $emp->name." ". $emp->patronymic."<br>"
?>
```

Как видно из листинга 2.9, при обращении к члену класса используется оператор `->`, после которого следует ввести имя этого члена. Заметьте, что при обращении к членам класса не добавляется символ `$`.

Обращение к закрытому члену класса `$age` завершится ошибкой **Fatal error: Cannot access private property employee::\$age**. На рис. 1.2 приводится схематическое изображение процесса доступа к членам объекта, снабженным разными спецификаторами доступа.

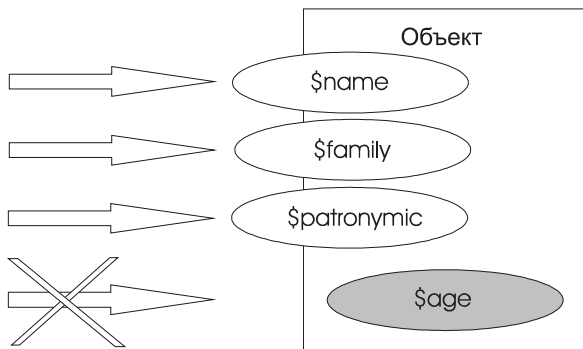


Рис. 1.2. Открытые и закрытые члены объекта

1.5. Методы класса. Член *\$this*

Популярность объектно-ориентированного подхода заключается в том, что классы представляют собой не просто контейнеры для хранения переменных (в качестве таких контейнеров в PHP могут выступать ассоциативные массивы), но также позволяют включать методы, обрабатывающие как открытые, так и закрытые члены класса.

Метод класса представляет собой обычную функцию PHP, которую предваряет один из спецификаторов доступа. В листинге 1.7 приводится пример класса `cls_mth`, в состав которого входит метод `show_message()`. Задача метода сводится к выводу в окно браузера сообщения **Hello world!**.

ЗАМЕЧАНИЕ

В листинге 2.12 можно опустить спецификатор доступа `public` перед методом `show_message()`. В отличие от других объектно-ориентированных языков, если в PHP не уточняется спецификатор, считается, что метод объявлен со спецификатором `public`. Именно потому, что в разных языках программирования используются разные подходы, не рекомендуется опускать спецификатор доступа при объявлении метода.

Листинг 1.7. Объявление метода класса

```
<?php
class cls_mth
{
    public function show_message()
    {
        echo "Hello world!";
    }
}

$obj = new cls_mth();

$obj->show_message();

?>
```

В листинге 1.7 метод класса не использует никаких членов, однако, как правило, методы вводят именно для обработки членов класса. Для обращения к любому элементу класса внутри метода следует применять конструкцию `$this->`. Член `$this`, который неявно присутствует в каждом классе, является ссылкой на текущий объект класса.

Вернемся к классу `employee`, определенному в листинге 1.5. Класс содержит закрытую переменную `$age`, определяющую возраст сотрудника. Создадим четыре метода:

- `get_age()` — метод, возвращающий значение закрытого члена `$age`;
- `set_age()` — метод, возвращающий значение закрытого члена `$age` и проверяющий его принадлежность к числовому типу данных и промежутку от 18 до 65;
- `get_info()` — метод, возвращающий фамилию и имя сотрудника;
- `get_full_info()` — метод, возвращающий фамилию, имя и возраст сотрудника.

ЗАМЕЧАНИЕ

Следует обратить внимание на то, что открытые методы и члены расположены в начале класса, а закрытые — в конце. Это негласное правило позволяет сосредоточить внимание программиста в первую очередь на открытом интерфейсе.

Листинг 1.8. Использование открытых методов для доступа к закрытому члену класса

```
<?php
class employee
{
    // Открытые члены
    public $surname;
    public $name;
    public $patronymic;

    // Открытые методы
    public function get_age()
    {
        return $this->age;
    }
    public function set_age($val)
    {
        $val = intval($val);
        if($val >= 18 && $val <= 65)
        {
            $this->age = $val;
            return true;
        }
        else return false;
    }
    public function get_info()
    {
        return $this->surname." ".$this->name." ".$this->patronymic;
    }
    public function get_full_info()
    {
        return "{$this->get_info()} ({$this->get_age()})";
    }

    // Закрытые члены
    private $age;
}
?>
```