УДК 004.438 PHP
ББК 32.973.26-018.1
Ф71

Фленов М. Е.

Ф71 PHP глазами хакера. — 4-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2019. — 256 с.: ил. — (Глазами хакера)

ISBN 978-5-9775-4062-9

Рассмотрены вопросы безопасности и оптимизации сценариев на языке PHP. Большое внимание уделено описанию типичных ошибок программистов, благодаря которым хакеры проникают на сервер, а также представлены методы и приведены практические рекомендации противостояния внешним атакам. Показаны реальные примеры взлома Web-сайтов и рекомендации, которые помогут создавать более защищенные сайты. В четвертом издании материал обновлен в соответствии с последней версией PHP 7, добавлено описание современных методов безопасности и защиты.

Для Web-программистов, администраторов и специалистов по безопасности

УДК 004.438 PHP
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Сависте*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн серии	*Марины Дамбиевой*
Оформление обложки	*Карины Соловьевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-4062-9

Оглавление

Предисловие к четвертому изданию

Это уже четвертое издание книги, и за время ее существования от первоначальной версии, похоже, ничего и не осталось. Все начиналось с небольшого сборника простых примеров с акцентом на безопасность, но от издания к изданию к ним постепенно добавлялись более сложные примеры, в том числе и по обеспечению безопасности, а сейчас весь этот материал еще и обновлен с учетом PHP 7-й версии. Надеюсь, мне удалось сделать книгу лучше.

Предыдущие издания книги комплектовались компакт-диском с примерами программного кода, рассматриваемого в книге, и различными дополнительными материалами. Однако сейчас компакт-диски такого свойства вышли из употребления, а Интернет стал настолько доступен, что скачать себе какие-то 200 килобайт не составляет проблемы. Поэтому все приведенные в книге исходные коды примеров вы сможете найти на сайте издательства «БХВ-Петербург» (см. *приложение*) или на моем сайте **www.flenov.info** в разделе **Книги**.

Предисловие

Эта книга посвящена одному из популярнейших языков программирования Web-сайтов — PHP. Вряд ли можно написать книгу, прочитав которую, вы сразу станете экспертом в такой области, как программирование, и моя книга тоже не претендует на подобную всеобъемлемость, но я постараюсь преподнести вам ее материал максимально просто и увлекательно, поскольку считаю, что информация лучше усваивается, когда она излагается в интересной форме, а вы, выполняя приведенные в книге примеры, видите результат своих действий.

Все книги по программированию, с которыми я знакомился, ставят перед собой цель научить читателя программировать, и только в конце книг их авторы обращают внимание на оптимизацию и безопасность. Мне кажется, что это не совсем верно, потому что если человека не научить программировать эффективно, то потом, с помощью пары финальных глав, переучить его будет сложно.

О безопасности нужно думать всегда, поскольку это проблема комплексная. Нет такого решения, которое может гарантировать, что после его реализации ваш код станет абсолютно безопасным. Вопросы встают разные, и снимать их приходится по-разному.

Ошибки, сделанные вами в процессе разработки программы, могут принести вам большую пользу (да!), потому что, выясняя их природу, вы сумеете лучше разобраться с возникшей ситуацией и найти правильное решение. Но это только в том случае, если ошибка закрадывается в программу на этапе ее разработки, и такая программа не попадает на рабочие серверы. Чтобы ошибка не оказалась фатальной для сайта и карьеры программиста, желательно постоянно отслеживать тенденции в технологиях компьютерной безопасности и проверять свой сайт на предмет возможных уязвимостей.

В этой книге описывается язык программирования PHP, начиная с самых его основ, и параллельно затрагиваются аспекты безопасности и оптимизации работы сценариев. Таким образом, вы с самого начала будете учиться создавать быстрые и защищенные приложения. О безопасности нужно думать всегда, а не в конце работы. Я даже скажу больше — начинать думать о ней следует еще до того, как вы начнете писать код.

Несмотря на то, что мы будем рассматривать безопасность и оптимизацию в течение всего периода изучения языка PHP, я не могу утверждать, что вы получите исчерпывающие знания. Очень многое останется за рамками книги, потому что нельзя предложить абсолютно эффективные и универсальные алгоритмы на все случаи жизни. Притом универсальность очень часто несовместима с понятиями эффективности и безопасности.

Помимо задач безопасности, мы затронем также вопросы создания сайтов, способных обрабатывать высокие нагрузки. Впрочем, для разработки сайта, который сможет выдерживать нагрузки типа Facebook, вам понадобится намного больше знаний, чем может дать эта книга, но я все же попробую дать вам хотя бы некоторые основы решения подобных проблем.

Рассматривая примеры того, как хакер может взломать сценарии, написанные на языке PHP, мы будем, как правило, подразумевать, что сервер работает под управлением ОС UNIX или одной из UNIX-подобных систем — например, Linux. Дело в том, что основная часть Web-сайтов с PHP-сценариями работает под управлением именно этих операционных систем. И для лучшего понимания материала вам необходимо иметь хотя бы поверхностное представление о какой-нибудь ОС семейства UNIX. А если вы знакомы с основами безопасности такой системы, наш разговор будет более продуктивным. Поэтому я рекомендую вам прочитать мою книгу «Linux глазами хакера» [1].

Использование Linux не обязательно. Эта книга и все примеры к ней написаны на компьютере под управлением macOS, которая, впрочем, своими корнями уходит в ту же Linux. Можно использовать даже Windows, потому что и под эту ОС тоже есть версия Web-сервера Apache, и на нее также можно установить базу данных MySQL и PHP — приложения, с которыми мы будем работать в процессе изучения материала книги.

Как вы уже, наверное, поняли, в этой книге будет рассматриваться самая популярная в Интернете связка — LAMP (Linux, Apache, MySQL и PHP), хотя первую букву в моем случае будет заменять M (macOS), — на этих четырех китах стоит большинство сайтов в Интернете, как и десять сайтов моей собственной разработки. За свою практику я создал много сайтов на этой платформе и считаю ее очень удачной для интернет-решений.

Ни одна книга не сможет сделать из человека хакера. Точно так же никакой труд не сделает из обезьяны человека — по крайней мере, не за одно поколение. Нужна долгая и кропотливая работа. В этой книге я не собирался научить кого-то хакерскому искусству. Если вы купили книгу только из-за этого слова, то, возможно, вас ждет разочарование. Но если вы приобрели ее для получения знаний о безопасности, то сможете найти для себя много нового и интересного. Я, по крайней мере, надеюсь на это и ставил перед собой такую задачу.

Об авторе

Меня зовут Михаил Фленов, и я увлекаюсь программированием с 1994 года. С 2009 года живу в Канаде недалеко от Торонто, где в течение восьми лет работал над различными проектами для американского офиса компании Sony. Вот один из

крупных проектов, в создании которого я принимал самое непосредственное участие: **www.sonyrewards.com** — сайт электронной коммерции и банк в одном флаконе. Есть в моем портфолио и сайт телепередачи «Wheel Of Fortune» (**www.wheeloffortune.com**) — по образу и подобию которой создано российское «Поле чудес» Леонида Якубовича. Эта передача до сих пор очень популярна в США, и трафик ее сайта весьма высок.

Разработаны так же мной и несколько сайтов с менее высоким трафиком. Среди них:

- **www.canada-area.com** — сайт о Канаде;
- **www.flenov.info** — мой персональный блог;
- **www.cweek.ca** — сайт о Канаде на английском языке.

Благодарности

Очень хочется поблагодарить всех тех, кто помогал мне в создании этой книги. Я не расставляю благодарности в порядке их значимости, потому что каждая помощь очень существенна для меня и для моей книги. Поэтому порядок не несет в себе никакого смысла, а выбран так, чтобы постараться никого не забыть.

Хочется поблагодарить издательство «БХВ-Петербург», с которым у меня сложилось уже весьма долгое и продуктивное сотрудничество. Надеюсь, что это сотрудничество не прервется никогда. Спасибо редакторам и корректорам за то, что исправляют мои недочеты и помогают сделать книгу лучше и интереснее.

Хочется поблагодарить мою семью, которая терпит исчезновения своего главы за компьютером. Я прекрасно понимаю, как тяжело видеть мужа и отца семейства, который вроде бы дома и в то же время отсутствует. Это напоминает загадку: «Висит груша, нельзя скушать». Я, правда, не груша и нигде не подвешен, но работать приходится много, и очень часто мой рабочий день длится 16 часов.

Хочу поблагодарить тех, кто разрешил тестировать свои серверы и сценарии в целях выявления ошибок, а также позволил просмотреть свои сценарии и настройки безопасности. Сотрудничество оказалось взаимовыгодным.

А единственная благодарность, которую я хотел бы принести раньше всех других и придать ей бóльшую значимость, — это вам, за то, что купили книгу, и моим постоянным читателям, которые также участвуют в создании моих книг. Все мои последние работы основываются на вопросах и предложениях читателей, с которыми я регулярно общаюсь через свой сайт **www.flenov.info**. Я постараюсь помочь вам по мере возможности и жду любых комментариев по поводу этой книги. Ваши замечания помогут мне сделать свою работу лучше.

На этом я заканчиваю вступление, и мы можем переходить к самому интересному в этой книге — к PHP, программированию и безопасности.

ГЛАВА 1

Введение в PHP

Программирование интернет-приложений, в том числе и работающих в браузере, предполагает большую ответственность за безопасность и защиту данных. Если вы напишете календарь, калькулятор или текстовый редактор, то такие программы, скорее всего, не привлекут внимания хакеров, да и не так уж много есть способов взломать систему, используя подобные приложения.

Но если вы разрабатываете Web-сайт, даже самый простой, ваш труд моментально окажется в зоне внимания злоумышленников. Один начинающий программист както возмутился: «Кому нужна моя личная домашняя страничка? Кто будет ее взламывать ради дефейса?» Однако внимание может привлечь не сам сайт или информация на нем, и дефейс (смена главной страницы) — далеко не единственная цель хакеров. Целью может быть не сам сайт, а сервер, на котором он работает. Получив полный доступ к сайту, можно использовать его в других, не вполне легальных целях.

Взломанные и подконтрольные злоумышленникам серверы в Интернете являются очень ценным ресурсом — ведь с их помощью можно проводить атаки на другие ресурсы Сети, причем анонимно. Через взломанные сайты распространяют вирусы и другую компьютерную заразу, рассылают спам и атакуют другие серверы.

Я считаю, что безопасность при разработке Web-сайтов важна вне зависимости от размера сайта и его привлекательности для злоумышленников, и мы будем думать о ней постоянно на протяжении всей книги. Да, безопасность — это комплексный вопрос, ответ на который кроется в особенностях не только языка программирования, но и сервера баз данных, и ОС, под управлением которой все это работает. Безопасность невозможно измерить, поэтому очень сложно сказать, когда безопасность становится достаточной.

1.1. Что такое PHP?

Язык PHP (один из вариантов расшифровки этой аббревиатуры: Personal Home Page Tools, инструменты персональных домашних страниц) — это язык сценариев с открытым исходным кодом, встраиваемых в HTML-код и выполняемых на Web-

сервере. PHP написан Web-разработчиками и для Web-разработчиков и является конкурентом таких продуктов, как Microsoft Active Server Pages (ASP) и Java Server Pages.

Сам по себе Web-сервер не умеет выполнять сценарии PHP — для этого необходима *программа-интерпретатор*. Такие интерпретаторы существуют для всех популярных Web-серверов (IIS, Apache) на всех основных платформах (Windows, Linux, macOS и пр.).

Язык PHP очень часто рассматривают в совокупности с Apache Web Server. Это бесплатный Web-сервер, который является лидером в своей области и используется более чем на половине серверов в Интернете (точную цифру назвать сложно, но любые данные указывают на превосходство этого сервера).

PHP позволяет встраивать фрагменты кода непосредственно в HTML-страницы, а интерпретированный код вашей страницы отображается пользователю. Код на языке PHP можно воспринимать как расширенные теги HTML, которые выполняются на сервере, или как маленькие программы, которые выполняются внутри страниц, прежде чем будут отправлены клиенту. Все, что делает код программы, незаметно для пользователя. Далеко не всегда это свойство является преимуществом, разве что только для очень маленьких сайтов. В случае с крупными проектами логику все же лучше держать отдельно от представления.

Практически ни один более или менее крупный Web-сайт не может работать без *хранилища данных*. Для решения этой задачи можно использовать текстовые файлы на сервере или базы данных (второе намного удобнее при обработке). В этой книге мы будем весьма часто рассматривать работу баз данных, а в качестве основной использовать самую распространенную на нынешний момент — MySQL. Это реляционная база данных с открытым исходным кодом, которая проста в использовании и поддерживается большинством хостинговых компаний.

Почему именно PHP и почему вообще вы должны изучать его? Сейчас все уходит в Сеть. Если лет десять назад пользователям приходилось для работы устанавливать соответствующие программы на локальный компьютер, то сейчас многие операции выполняются из браузера. Мы в браузере работаем с почтой, документами и электронными таблицами, и в нем большинство из нас проводит больше всего времени.

В Канаде сейчас PHP — один из самых востребованных языков. Здесь стабильный и высокий спрос на специалистов со знанием PHP, хотя в среднем .NET- и Java-программисты зарабатывают все же больше. Но зато PHP-программисту проще найти работу. Моя первая работа в Канаде как раз и была связана с этим языком программирования.

1.2. Создание сайта для Apache

Код PHP — это не просто HTML-разметка, которая выполняется прямо в браузере. Это код, который работает на сервере. Самым популярным Web-сервером сейчас является Apache, и здесь я покажу вариант его настройки. В качестве операционной

системы мы воспользуемся macOS, но настройка Apache в Linux-системах почти не отличается — разница может быть только в расположении конфигурационных файлов. К тому же, у меня есть целая книга по Linux, и там эта тема подробно раскрыта, так что я могу порекомендовать вам с ней ознакомиться (имеется в виду уже упомянутая в *предисловии* книга «Linux глазами хакера»).

В macOS конфигурационные файлы Apache расположены в каталоге /etc/apache2. Причем по умолчанию конфигурация распределена по множеству файлов. В некоторых системах наоборот — по умолчанию все хранится в одном большом файле http.conf. Например, у меня на выделенном сервере в интернет-хостинге стоит ОС CentOS, и там по умолчанию обстоят дела именно так.

Способ расположения конфигурационных файлов можно изменить, т. к. сервер Apache весьма гибкий, и его можно конфигурировать по-разному. Но это тема отдельной книги по Linux или Apache.

Итак, для начала открываем файл httpd-vhosts.conf в любом текстовом редакторе (я предпочитаю редактор vi):

```
vi /etc/apache2/extra/httpd-vhosts.conf
```

и добавляем следующую секцию:

```
<VirtualHost *:80>
    DocumentRoot "/Users/michaelflenov/Projects/Web/phpbook"
    ServerName phpbook
    ErrorLog "/var/log/apache2/phpbook.com-error_log"
    CustomLog "/var/log/apache2/dummy.access_log" common
</VirtualHost>
```

Здесь я создаю новый виртуальный хост, файлы которого будут располагаться в каталоге /Users/michaelflenov/Projects/Web/phpbook. Именем сервера (параметр `ServerName`) будет `phpbook`, и именно по этому имени в браузере мы сможем обратиться к нашему сайту.

Параметры `ErrorLog` и `CustomLog` не обязательны с точки зрения работы сайта, но желательно обеспечить их уникальность — для каждого сайта индивидуальные, чтобы проще было отлаживать возможные проблемы.

Чтобы Apache прочитал настройки, его нужно перезапустить, выполнив команду:

```
sudo apachectl restart
```

В данном случае `sudo` указывает на то, что команду нужно выполнять с правами администратора. Управление сервером — достаточно привилегированная задача, и ее нельзя доверить абсолютно каждому, поэтому в Linux и в macOS для этого требуются привилегированные права. Как дела обстоят в Windows, я не знаю — я не ставил Apache на эту ОС уже лет пятнадцать, а это, кажется, были времена еще Windows 95.

Команда, которая управляет Web-сервером, это `apachectl`. А параметр `restart` говорит о том, что нужно перезагрузить сервер, и в этот момент будет перезагружена конфигурация.

Сайт готов, но браузер пока не сможет загрузить его по имени phpbook, потому что не знает, что это имя должно указывать на ваш же компьютер. Обычно, когда мы обращаемся к сайтам в Интернете, вопросом превращения имени в адрес занимается сервер DNS, но в нашем случае его нет. Впрочем, при локальной разработке можно обойтись и файлом hosts. Дело в том, что когда мы в браузере вбиваем какое-либо имя, то сначала система ищет адрес этого имени в файле hosts, и если там не найдет его, то обращается к DNS-серверам в Интернете.

В Linux и macOS этот файл находится здесь: /etc/hosts, и в него мы добавляем новую запись:

```
127.0.0.1   phpbook
```

Вот теперь все готово, чтобы мы смогли работать с PHP-скриптами и тестировать их с помощью нашего сайта. Некоторые любят задействовать программы типа MAMP, которые предоставляют визуальный интерфейс для конфигурирования локальных сайтов. Честно говоря, я не вижу в этом особого смысла, потому что вручную (без дополнительной программной обертки) это делается очень даже просто.

Так как в файле hosts мы указали просто имя — без имени домена типа .ru, то, если просто вбить в адресной строке браузера phpbook, браузер может запустить по этому слову поиск Google, а не сайт, поэтому нужно вводить: http://phpbook/. Чтобы не усложнять себе задачу, вы можете добавить в конфигурацию и в файл hosts любой домен на свой выбор — например, заменить везде phpbook на phpbook.edu.

1.3. Как работает PHP?

Что значит «встраиваемый» язык? Рассмотрим простой пример кода Web-страницы, в которой используются PHP-инструкции (листинг 1.1).

Листинг 1.1. Код страницы с PHP-инструкциями

```
<html>
<head>
<title> test page </title>
</head>

<body>
<?php
$title='We are glad to see you again';
?>
<p>hello. <?php echo $title ?>
<p>current time <?php echo date('y-m-d h:i:s') ?>
</body>
<html>
```

Инструкции PHP по умолчанию заключаются в тег <?php ... ?>. Мы пока не будем вникать в код, показанный здесь, потому что сейчас главная задача — уяснить об-

щий принцип работы (уже прочитав следующую главу, вы поймете написанный здесь код). Если теперь вы загрузите эту страничку с Web-сервера, то должны увидеть примерно то, что изображено на рис. 1.1.

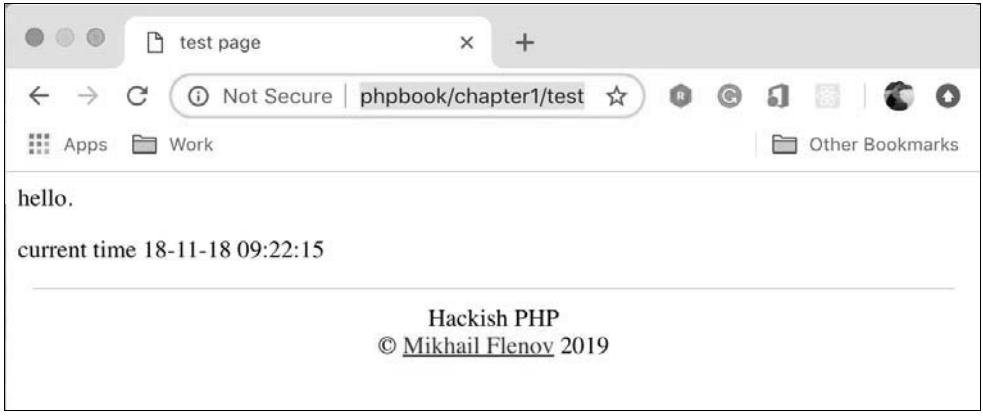

Рис. 1.1. Страница, в которой работает код PHP

Я живу в Канаде, и мой MacBook имеет канадские настройки, поэтому и дата на иллюстрации выглядит таким странным образом: **18-11-18**. На самом деле это 18 ноября 2018 года, и скоро Рождество. В предыдущем издании на соответствующей иллюстрации стояла дата **15-12-15**, что было к Рождеству еще ближе.

Наш сайт настроен так, что корень его находится в каталоге /Users/michaelflenov/Projects/Web/phpbook (см. *разд. 1.2*). Чтобы загрузить тестовый файл, я сохранил его код в файл с именем test.php в каталоге /Users/michaelflenov/Projects/Web/phpbook/chapter1.

Теперь, если ввести в адресной строке браузера URL `http://phpbook/chapter1/test.php`, Web-сервер исполнит PHP-код, расположенный между тегами `<?php` и `?>`, и возвратит браузеру уже обработанный код.

Давайте посмотрим на исходный код страницы в браузере. Для этого выберите в его меню **Вид | В виде HTML** (View | Source) — само меню в зависимости от браузера может отличаться. Перед вами откроется окно текстового редактора, в котором будет содержаться примерно следующий код:

```
<HTML>
<HEAD>
<TITLE> Test page </TITLE>
</HEAD>

<BODY>
<p>Hello. We are glad to see you again
<p>Current time 15-12-15 07:03:45
</BODY>
<HTML>
```

Как видите, никаких PHP-инструкций больше нет. Все они исчезли. Это связано с тем, что сервер обработал наши команды и передал пользователю чистый HTML-код. Можно также сказать, что пользователь видит лишь результат работы сценария. Когда пользователь запрашивает страницу, сервер обрабатывает все PHP-инструкции на этой странице и возвращает только результат обработки на чистом HTML.

Так мы создаем простую HTML-страницу, в которой присутствует код PHP. В некоторых языках, схожих по назначению с PHP (например, в Perl), происходит обратное. Там вы пишете код программы, а для добавления HTML-тегов нужно писать специальные инструкции. В PHP возможны оба способа.

Мы будем использовать много чистого PHP-кода вместе с HTML, но для реальных проектов это все же нежелательно. Исполняемый код (PHP) должен быть максимально отделен от представления (HTML). Так удобнее и эффективнее с точки зрения сопровождения. Это большая и интересная тема, и к ней мы будем неоднократно возвращаться.

Язык PHP является *интерпретируемым*. Это значит, что его не надо компилировать в программу (хотя можно и откомпилировать), а выполнение происходит «на лету». Это очень удобно, но иногда может приводить к издержкам, потому что интерпретация текста в машинный код отнимает процессорное время. Впрочем, интерпретатор PHP умеет компилировать «на лету»: во время первого чтения файла он интерпретируется, а если потом происходит обращение к уже использованному ранее коду, то он просто исполняется без повторной компиляции.

Есть у интерпретируемости и еще один недостаток — сценарии PHP поставляются в исходных кодах, и, если вы будете их распространять, любой программист сможет увидеть ваш труд и использовать в своих целях. Пользователи же вашего сайта не смогут увидеть исходный код, потому что в браузер попадает лишь результат или HTML-документ.

ГЛАВА 2

Основы PHP

В этой главе мы познакомимся с основами языка PHP и научимся писать простейшие сценарии. Нам предстоит заложить фундамент, на котором будет построен процесс изучения всего материала книги. Уже в этой главе мы овладеем некоторыми приемами, которые помогут сделать ваш код лучше.

Даже если вы знакомы с PHP, я советую вам просмотреть эту главу. Возможно, вы почерпнете для себя что-то новое. В любом случае я считаю полезным взглянуть на знакомые вещи со стороны. Конечно же, опытному программисту я мало чего нового расскажу об основах PHP, но некоторые советы из моего личного опыта могут оказаться и ему интересными и полезными.

Итак, нам предстоит узнать, как пишутся инструкции PHP, что такое переменные, мы познакомимся с логическими операторами и обсудим, как можно управлять выполнением программы.

Я постараюсь сделать описание максимально увлекательным, хотя на данном этапе это непросто. Неподготовленный читатель должен сначала усвоить необходимые основы, без которых мы не сможем начать писать более интересный код для Сети. А так как примеров поначалу будет мало, описание может показаться сухим и занудным. Но я сделаю все, чтобы вы не заскучали.

Чтобы материал лучше запоминался и его интереснее было читать, я рекомендую вам самостоятельно создавать файлы сценариев и проверять результат работы. Только так вы приобретете достаточный опыт и лучше запомните излагаемый в книге материал.

Что ж, приступаем к более глубокому знакомству с одним из самых интересных и мощных языков программирования Web-сайтов — с языком PHP.

2.1. PHP-инструкции

Как мы уже знаем, PHP-инструкции пишутся прямо в HTML-документе. Но как Web-сервер определяет внутри HTML-документа PHP-код, который надо выполнить, а какой просто вернуть браузеру? Как отделить PHP от HTML? Очень просто.

С помощью специального соглашения вы указываете, где начинается и заканчивается код. Все остальное воспринимается как HTML-документ.

Чаще всего для обозначения начала и конца PHP-инструкций используется следующий формат (потому что он установлен по умолчанию):

```
<?php
код PHP
?>
```

Все, что располагается между тегами `<?php` и `?>`, воспринимается сервером как PHP-код и обрабатывается соответствующим образом. Остальное воспринимается как HTML-документ, передается клиенту без изменений и обрабатывается уже в браузере пользователя.

Такой формат наиболее предпочтителен, и в этом случае вы можете быть уверены, что он будет корректно обработан сервером. Однако допускается использовать и другие варианты, которые поддерживаются в настоящий момент. Правда, вы должны отдавать себе отчет, что только теги `<?php` и `?>` будут гарантированно поддерживаться во всех версиях. Именно так заявили разработчики много лет назад. Короткие варианты (см. далее) тоже работают и, скорее всего, будут работать всегда.

Теперь рассмотрим другие варианты выделения PHP-кода. Самый короткий и популярный такой:

```
<?
код PHP
?>
```

Чтобы сервер распознал эту форму записи, следует включить ее поддержку со стороны PHP. Для этого нужно скомпилировать интерпретатор PHP с опцией:

```
--enable-short-tags
```

или в файле настроек PHP php.ini изменить значение параметра `short_open_tag` на `on`.

На мой взгляд, этот формат оформления PHP-кода стал самым популярным, и я сам в большинстве случаев использую сейчас именно его, хотя раньше и рекомендовал отключать короткие теги. Предполагаю, что этот формат тоже будет жить долго.

Следующий тип записи выглядит так:

```
<%
код PHP
%>
```

Этот вариант оформления кода немного проще, чем первый, но именно он принят в ASP.

А самый громоздкий способ выглядит так:

```
<SCRIPT LANGUAGE="php">
код PHP
</SCRIPT>
```

Давайте попробуем что-нибудь написать на языке PHP. Первое, с чего можно начать, — напечатать какой-либо текст и узнать информацию об установленном на сервере интерпретаторе PHP. Для этого создайте файл information.php со следующим содержимым (листинг 2.1).

Листинг 2.1. Вывод информации об интерпретаторе PHP

```
<html>
<head>
<title> Test page </title>
</head>

<body>
<?php
print("<p>This is information about php</p>");
phpinfo();
?>
</body>
<html>
```

Загрузите с помощью FTP-клиента этот файл на ваш Web-сервер или поместите его в каталог сервера на локальном диске, если вы используете локальную версию сервера. Теперь загрузите файл с помощью браузера. Если вы поместили файл в корневой каталог сервера, то в адресной строке браузера нужно набрать `http://phpbook/chapter2/l21.php`, где `phpbook` — имя вашего сервера. Результат выполнения сценария можно увидеть на рис. 2.1.

Такой сценарий чаще всего используют для тестирования работоспособности сервера, правильности установки PHP и получения параметров. Рассмотрим, из чего состоит этот сценарий.

Между тегами `<?php` и `?>` расположены две строчки команд:

```
print("<p>This is information about php</p>");
phpinfo();
```

В первой строчке вызывается функция `print()`. С ее помощью можно выводить в окно браузера какой-нибудь текст. Как мы уже знаем, для вывода простого текста достаточно использовать HTML, но в нашем случае я хотел показать, как это делается посредством PHP-функций. В дальнейшем мы познакомимся с этой функцией более подробно, а пока достаточно будет основных сведений.

Любая PHP-функция может получать параметры, которые указываются в скобках после имени функции. Для функции `print()` в скобках нужно задать текст, который должен быть напечатан в браузере клиента. Текст заключается в двойные или одинарные кавычки. Посмотрите на текст, который мы указали, и вы увидите, что в нем есть HTML теги `<p>`, которые используются для оформления параграфов. Текст может идти вперемежку с HTML-тегами, и его можно форматировать.

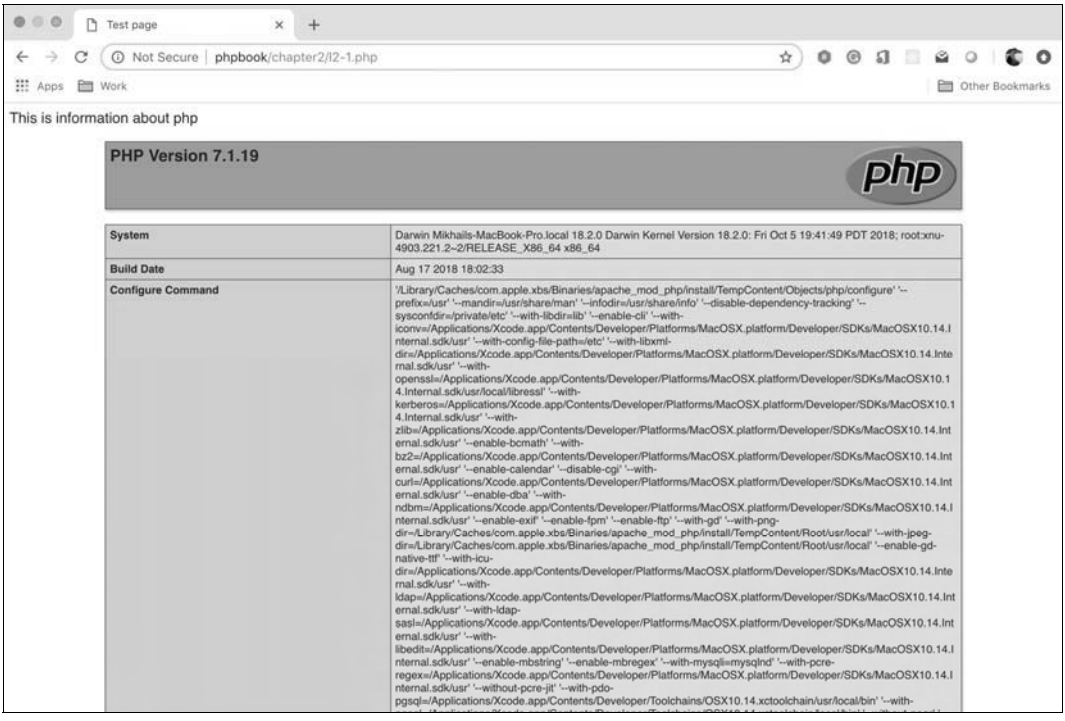

Рис. 2.1. Результат вывода информации о PHP

Не знаю, есть ли такое правило, но лично я не рекомендую использовать функцию `print()` для вывода текста. В реальных условиях с ее помощью следует выводить значения переменных. Если же нужно вывести текст, то выносите его за пределы PHP-кода — например, так:

```
<?php
?>
<p>This is information about php</p>
<?php
phpinfo();
?>
```

Здесь сначала открыт один блок кода PHP, в котором ничего нет, но допустим, что там есть какая-то логика. Потом этот блок закрывается, и выводится простой HTML-заголовок, а дальше снова открывается еще один блок кода, где мы продолжаем писать на PHP.

То есть я рекомендую размещать HTML-код за пределами PHP-блоков. А с помощью функции `print()` лучше выводить только переменные и, если нужно, небольшие сообщения.

Далее у нас в коде вызывается функция `phpinfo()`. Она отображает в браузере отформатированную информацию о PHP, которую вы и можете наблюдать (см. рис. 2.1). Для этой функции не нужны параметры, поэтому в скобках ничего не указано.

При создании страниц вы в любой момент можете переключаться между режимами PHP и HTML. Точнее сказать, что в любое место HTML-документа можно вставлять PHP-команды, как показано в листинге 2.2.

Листинг 2.2. Пример переключения между HTML и PHP

```
<html>
<head>
<title> Vision </title>
</head>
<body>
<p> Hello </p>
<p> <?php $i = 1; print("this is    ");?> </p>
<p> i = <?php print($i) ?> </p>
</body>
<html>
```

В этом примере две строки содержат PHP-код, а остальные — HTML-код. Вы можете вставлять участки кода на языке PHP так часто, как вам это нужно.

Но это еще не все. В первой строке PHP-кода мы объявляем переменную `$i`, которая будет равна 1. Что такое переменная? Сейчас вам достаточно понимать, что это ячейка (или область) памяти для хранения определенных значений (чисел, строк и т. д.), с которыми потом можно производить вычисления или другие действия. *Переменные* — это как бы маленькие чемоданчики с именами внутри памяти компьютера, в которые можно запрятать данные. Имена переменных начинаются с символа `$`. Итак, наша ячейка памяти будет называться `$i` и содержать значение 1.

Во втором участке кода происходит печать содержимого ячейки памяти с именем `$i`.

Как видите, значение ячейки памяти `$i` никуда не исчезло и равно 1, несмотря на то, что переменная создана в одном месте, а использована в другом. Таким образом, переменные сохраняют свои значения на протяжении всего документа.

Есть более короткий метод вывести значение переменной:

```
<?= ПЕРЕМЕННАЯ ?>
```

Вместо слова `ПЕРЕМЕННАЯ` можно указывать имя переменной, и PHP просто выведет значение переменной. Приведенный ранее код мы могли бы переписать с использованием этого короткого варианта следующим образом:

```
<p> Hello </p>
<p> <?php $i = 1; print("this is    ");?> </p>
<p> i = <?= $i ?> </p>
```

На мой взгляд, такой вариант выглядит весьма элегантно.

2.2. Подключение файлов

Многократное использование кода — вечная проблема для любого программиста. Когда мы разрабатываем новый проект, нам абсолютно неинтересно решать те же проблемы, которые были уже решены при реализации предыдущих. Было бы хорошо просто воспользоваться имеющимся кодом и поддерживать его только в одном месте.

Вы, наверное, не раз слышали о динамических библиотеках Windows. Это библиотеки, в которых хранятся различные ресурсы (картинки, значки, формы диалоговых окон, меню и другие типы ресурсов) и/или код программы. Любая программа может загрузить такую библиотеку и использовать ее содержимое. Например, OpenGL — графическая библиотека, в которой хранятся функции для создания 3D-графики практически любой сложности. Программист может загрузить эту библиотеку в память компьютера и использовать в своих проектах. Соответственно, ему не придется для каждой программы заново писать код графических функций, а можно будет воспользоваться тем, что уже хорошо сделано другими разработчиками.

При программировании Web-страниц проблема многократного использования кода становится еще более острой. Ваш сайт может содержать сотню файлов с кодом, и писать в каждом из них одни и те же команды неудобно и нерационально. Файлы будут получаться объемными, а сопровождение большого количества файлов превратится в сущий ад.

Многократное использование кода в PHP реализовано через *подключение файлов*. Подключение файлов можно воспринимать и как еще один способ внедрения PHP-кода в Web-страницу. Осуществляется оно с помощью функций `include()` и `require()`, имеющих четыре разновидности:

```
include('/filepath/filename');
include_once('/filepath/filename');

require('/filepath/filename');
require_once('/filepath/filename');
```

Все эти функции подключают указанный в скобках файл. В предыдущих версиях между функциями `include()`/`include_once()` и `require()`/`require_once()` присутствовала небольшая разница в скорости работы. В настоящее время существует разница только в генерируемых ошибках. Первые две функции при обнаружении ошибки (например, отсутствует подключаемый файл) выдают предупреждение и продолжают выполнение сценария. Функции `require()`/`require_once()` выдают сообщение о критической ошибке, и дальнейшая работа прерывается.

Различие между `include()`/`require()` и `include_once()`/`require_once()` состоит в том, что вторая пара функций гарантирует подключение указанного файла к текущему файлу только один раз. Иногда это бывает необходимо, чтобы дважды не подключать файл с критически важным кодом, который должен быть только в единствен-