2-е издание

CGI программирование

на Perl

Символ

O'REILLY®

Scott Guelich, Shishir Gundavaram & Gunther Birznieks

CGI Programming with Perl

Second Edition

Scott Guelich, Shisbir Gundavaram and Gunther Birznieks

O'REILLY®

CGI программирование на Perl

Второе издание

Скотт Гулич, Шишир Гундаварам,
Гюнтер Бирзнекс

Санкт-Петербург
2001

Скотт Гулич, Шишир Гундаварам, Гюнтер Бирзнекс

CGI программирование на Perl

Перевод Т. Морозовой

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*А. Михайлов*
Редактор	*А. Кузнецов*
Корректура	*С. Журавина*
Верстка	*Н. Грибещенко*

Гулич С., Гундаварам Ш., Бирзнекс Г.

CGI программирование на Perl. – Пер. с англ. – СПб: Символ-Плюс, 2001. – 480 с., ил.

ISBN 5-93286-016-2

Эта книга – отличное начало для тех, кто хочет научиться писать CGI-программы, обеспечивающие вывод динамически изменяемых данных на веб-сайте, и уже немного знаком с языком Perl, пользующимся большой популярностью среди веб-разработчиков. Данное издание, в основу которого положен бестселлер «CGI программирование в WWW», полностью переписано с целью познакомить читателей с современными технологиями, доступными благодаря модулю CGI.pm и последним версиям языка Perl.

В книге приводятся примеры создания высокопроизводительных и безопасных CGI-приложений, подробно описывается модуль CGI.pm, дан обзор протокола HTTP, обсуждается применение JavaScript для обработки форм, работа с базами данных, вывод динамической графики, создание поисковой системы и системы на основе XML, а также многое другое. Издание послужит прекрасным руководством и незаменимым справочником. Содержащийся в нем материал позволит вам стать хорошим CGI-разработчиком.

ISBN 5-93286-016-2
ISBN 1-56592-419-3 (англ)

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
тел. (812) 324-5353. Лицензия ЛП N 000054 от 25.12.98.

Подписано в печать 15.02.2001. Формат 70x100 $^{1}/_{16}$. Печать офсетная.
Объем 30 печ. л. Тираж 3000 экз. Заказ N 43.
Отпечатано с диапозитивов в ФГУП «Печатный двор» Министерства РФ
по делам печати, телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

Оглавление

Предисловие

Первое издание книги «CGI-программирование для World Wide Web» вышло в начале 1996 года. Сеть тогда была другой: число веб-узлов не превышало 100 000, Netscape Navigator 2.0 (первый броузер, совместимый с JavaScript™) только что появился, а язык Java™ существовал менее года и использовался преимущественно для апплетов. Оставаясь молодой, Сеть стремительно развивается.

В 1996 году единственным проверенным и понятным способом создания динамических страниц в сети был CGI. Но, тем не менее, лишь на некоторых сайтах был использован весь его потенциал. В первом издании книги Шишир писал:

> Пользователи современных компьютеров ожидают получить привычные ответы на особые вопросы. Прошли те дни, когда людям было достаточно получить один ответ для всех, составленный сотрудниками компьютерного центра. Напротив, теперь любой продавец, менеджер и инженер хочет увидеть подходящий ответ на любой, даже специфичный, запрос. И если это можно сделать хотя бы на одном компьютере, то почему бы и не в Сети?
>
> Это задача для CGI. Для персонала можно с помощью понятных диаграмм отображать продажи по отдельным товарам за каждый месяц. А покупатель сможет вводить ключевые слова, чтобы найти информацию о требуемой продукции.

В 1996 году это было смелое заявление. Сегодня все это обычно для бизнеса. В этом смысле CGI оправдал возлагаемые на него надежды.

Эта книга не только о том, как писать CGI-сценарии. Она о программировании для Сети. И хотя в центре нашего внимания будет CGI-программирование на Perl (поэтому изменилось и название книги в этом издании), многое из рассмотренного также справедливо для всех веб-разработок на стороне сервера. И даже если вы работаете с менее прогрессивными технологиями, попытка изучить CGI не будет лишней и окупится впоследствии.

Что есть в этой книге

Поскольку за последние несколько лет CGI изменился очень сильно, можно только надеяться, что это издание охватит все изменения. Большая часть книги написана заново. Добавлены новые темы, среди них: обзор модуля CGI.pm, шаблоны HTML, безопасность, JavaScript, XML, поисковые системы, стили, совместимые высокопроизводительные альтернативы CGI. Расширены и обновлены разделы первого издания, посвященные управлению сессиями, электронной почте, динамическим изображениям и реляционным базам данных. Наконец, мы начали книгу с обзора протокола HTTP – основы сети. Поняв HTTP, гораздо проще разобраться с CGI.

Несмотря на все это, цель книги осталась прежней: в ней есть все, что нужно вам для того, чтобы стать хорошим CGI-разработчиком. Это не книга типа «учись на примерах», где есть только CGI-сценарии и описание их работы (таких книг о CGI уже достаточно). И хотя эти книги тоже полезны, особенно если примеры в них соответствуют вашей задаче, они показывают, *как* можно что-то сделать, не объясняя *зачем*. Цель этой книги – научить вас основам CGI, чтобы вы могли создавать свои собственные CGI-сценарии для решения любой задачи. Не волнуйтесь, в книге хватает и примеров – они служат для того, чтобы иллюстрировать рассказ.

Надо признать, что в этой книге мы делаем акцент на Unix. И Perl и CGI первоначально были созданы на платформе Unix, поэтому Unix остается самой популярной платформой для Perl- и CGI-разработок. Разумеется, поддержка Perl и CGI есть и в других системах, включая популярные 32-разрядные системы, созданные Microsoft: Windows 95, Windows 98, Windows NT и Windows 2000 (далее мы будем называть их *Win32*). В основном мы будем говорить о Unix, но будут упомянуты и важные моменты для работы в системах, отличных от Unix.

Во всех примерах мы используем веб-сервер Apache по нескольким причинам: это самый популярный на сегодняшний день веб-сервер, доступный для большинства платформ, бесплатный, соответствующий принципу «open source» и поддерживающий модули (такие как *mod_perl* и *mod_fastcgi*), которые увеличивают мощь и производительность Perl как языка для веб-разработок.

Что вы должны знать

Мы надеемся, что вы уже неплохо разбираетесь в Perl. Хотя в первом издании книги обсуждались и другие языки программирования, в этом издании книги речь пойдет только о Perl. CGI поддерживает многие языки программирования, но Perl стал предпочтительным выбором.

Тем, кто еще не знаком с Perl, поможет отличная книга Randal Schwartz и Tom Kristiansen *Learning Perl*, Second Edition, издательство O'Reilly & Associates, Inc[1]. Мы рекомендуем после изучения основ Perl прочитать книгу Larry Wall, Jon Orwant *Programming Perl*, Third Edition того же издательства[2]. Это справочник, ставший стандартным для Perl-разработчиков. Другие ресурсы по Perl перечислены в приложении A.

Мы обсудим многие модули от CPAN (Comprehensive Perl Archive Network). Если вы никогда раньше не загружали и не устанавливали модули с CPAN, обратитесь к приложению B.

Кроме того, вы должны быть знакомы с *perldoc*, стандартным инструментом для просмотра документации, полезным по двум причинам. Во-первых, он обеспечивает доступ к удобной и обширной документации, распространяемой вместе с Perl. Во-вторых, модули, полученные со CPAN, очень просто использовать. Описание *perldoc* также можно найти в приложении B.

Обзор книги

В главе 1 рассмотрена общая информация о CGI, включая историю, конфигурирование веб-сервера и простой CGI-сценарий.

Главы 2–4 посвящены основам использования CGI. Мы начнем с обзора протокола HTTP и покажем, как CGI строится на нем. Затем мы рассмотрим HTML-формы – основной способ передачи данных в CGI-сценарии.

В главах 5 и 6 рассматриваются популярные модули, позволяющие писать CGI-сценарии проще. Кроме того, мы изучим различные подходы к созданию динамических страниц.

В главе 7 мы расскажем, как иная технология – JavaScript при использовании с CGI-сценариями позволяет реализовывать более мощные решения.

В главах 8–13 представлены основные задачи, которые решаются при помощи CGI, такие как безопасность, хранение постоянных данных и отслеживание пользователей, посещающих страницы, а также специальные темы, такие как отправка электронной почты, реализация поиска по сайту и создание динамических изображений.

В главе 14 речь идет о XML, обеспечивающем интерфейс с другими информационными серверами.

---

[1] Р. Шварц, Т. Кристиансен «Изучаем Perl», издательство «BHV-Киев».

[2] Л. Уолл, Т. Кристиансен, Р. Шварц «Программирование на Perl», 3 издание, издательство «Символ-Плюс», июнь 2001 г.

В главах 15–17 мы объясним, как лучше писать CGI-сценарии, применяя различные стратегии отладки сценариев, обсудим правила написания хорошего кода и расскажем, как улучшить производительность сценариев.

В приложении A приведен список ресурсов, где можно найти информацию о CGI.

В приложении B описано, как загрузить данные со CPAN.

Принятые соглашения

`Моноширинным шрифтом`

выделены HTTP-заголовки, коды состояния, типы содержимого MIME, директивы в конфигурационных файлах, массивы, операторы, имена переменных (за исключением примеров) и текстовые результаты вывода.

Курсивом

выделены имена файлов, пути каталогов, названия групп новостей, Интернет-адреса (URL), адреса электронной почты, описываемые термины, команды, параметры/ключи, имена программ и подпрограмм, функции, методы и имена узлов.

ПРОПИСНЫМИ БУКВАМИ

выделены переменные окружения, HTML-атрибуты и HTML-теги (внутри угловых скобок < >).

Как с нами связаться

Мы тестировали и проверяли всю информацию, приведенную в книге, но может оказаться, что некоторые возможности изменились или что некоторые ошибки появились уже при выпуске книги. Пожалуйста, сообщайте нам об ошибках, которые вы обнаружите, а также присылайте предложения для будущих изданий по следующему адресу:

O'Reilly&Associates, Inc.

101 Morris St.

Sebastopol, CA 95472

(800) 998-9938 (для США и Канады)

(707) 829-0515 (международный/местный)

(707) 829-0104 (факс)

1

Начало

Как и все в Интернете, общий шлюзовый интерфейс (Common Gateway Interface, CGI) прошел в своем развитии очень большой путь за очень короткое время. Всего несколько лет назад CGI-сценарии были скорее новинкой, чем полезным инструментом; с их помощью создавались счетчики посещений и гостевые книги, в основном, любителями. Сегодня CGI-сценарии пишут профессиональные веб-разработчики, они и обеспечивают логику в большой структуре, какой стал Интернет.

История

Хотя Интернет стал очень популярен именно теперь, он не так уж нов. Предшественником современного Интернета стала сеть ARPAnet, созданная 30 лет назад в учебных целях в министерстве обороны США. Первые 25 лет Интернет развивался постепенно, но внезапно произошел скачок.

В Интернете всегда существовало множество протоколов для обмена информацией, но когда появились броузеры NCSA Mosaic и затем Netscape Navigator, они вызвали удивительный рост. В последние 6 лет число веб-сайтов выросло с тысячи до более чем 10 миллионов. Сегодня, слыша термин «Интернет», большинство людей представляет себе веб-страницы. Другие протоколы передачи данных, электронная почта, FTP, чаты и каналы новостей по-прежнему остаются популярными, но они становятся в WWW вторичными, поскольку многие пользуются веб-сайтами как шлюзами для доступа к другим службам.

Разумеется, WWW – не первая технология, позволившая публиковать информацию и обмениваться ей, но в ней было что-то особенное, что вызвало ее мгновенный рост. Хочется сказать, что именно интерфейс CGI (а не FTP и Gopher) вызвал рост веб-сервиса в Интернете. Но это было бы неправдой. Вероятно, веб-сервис поначалу стал популярен из-за того, что он был «с картинками»: он создавался так, чтобы представлять различные виды информации. Практически с самого начала броузеры могли показывать изображения; HTML поддерживал управление выводом, что позволяло проще читать и размещать информацию. Эти возможности продолжали развиваться, в Netscape добавилась поддержка новых расширений HTML, входившая в каждый новый выпуск броузера.

Таким образом, первоначально Интернет был набором домашних страниц и веб-сайтов, содержащих различную информацию. Но никто по-настоящему не знал, что с ними *делать*, особенно деловые люди. В 1995 году для представителей корпораций были обычными слова «Разумеется, Интернет это хорошо, но многие ли заработали в онлайне?» Как быстро все изменилось!

Как CGI используется сегодня

Сегодня электронная коммерция взлетела, и слова «точка-com» появляются повсюду. В основе этого прогресса лежат несколько технологий, и CGI без сомнения одна из самых важных. CGI позволяет Web *делать* что-то, а не просто быть набором статических ресурсов. *Статический* ресурс – это то, что не меняется от запроса к запросу, таким является HTML-файл или графика. *Динамический* ресурс содержит информацию, которая может изменяться от запроса к запросу в зависимости от ряда условий, включая изменяющийся источник данных (например, база данных), пользователя, посылающего запрос, или введенные пользователем данные. Поддерживая динамическое содержимое, CGI позволяет опубликовывать на веб-серверах приложения, доступные любым пользователям на любых платформах со всего мира через стандартную клиентскую программу – веб-броузер.

Сложно перечислить все, что можно сделать при помощи CGI. Если вы хотите обеспечить поиск по своему сайту, то CGI-приложение – именно то, что нужно для обработки информации. Если вы заполняете форму на веб-сервере, то данные обрабатываются CGI-приложением. Если вы делаете заказ в электронном магазине, то CGI-приложение подтверждает действительность вашей кредитной карты и регистрирует операцию. Если вы наблюдаете за постоянно обновляемой диаграммой, то, скорее всего, рисует диаграмму именно CGI-приложение. Разумеется, в последние годы появились другие технологии, которые могут выполнять подобные задачи, о некоторых из них мы расскажем. Тем не менее CGI ос-

тается самым популярным инструментом для выполнения подобных задач.

Введение в CGI

CGI может так много, потому что это очень простой интерфейс: он содержит минимум, позволяющий создавать веб-страницы в зависимости от внешних процессов. Обычно, когда веб-сервер получает запрос статической страницы, он находит соответствующий HTML-файл в своей файловой системе. Когда веб-сервер получает запрос к CGI-сценарию, веб-сервер исполняет сценарий как другой процесс (например, отдельное приложение); сервер передает этому процессу параметры и получает результаты, которые затем возвращаются клиенту так, будто они получены из статического файла (рис. 1-1).

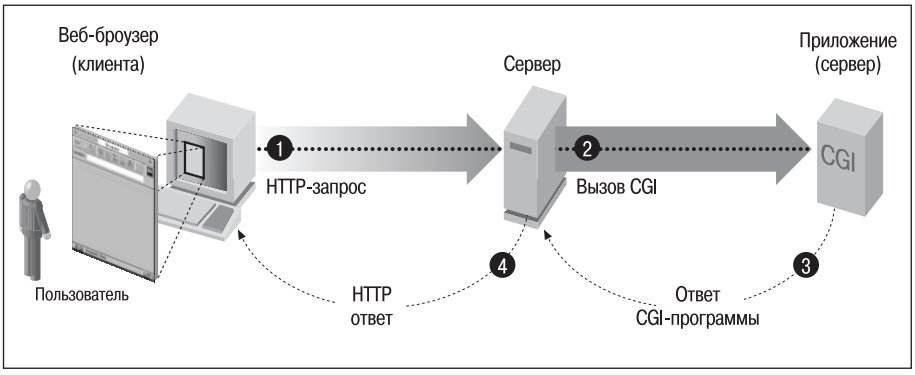

Рис. 1-1. Как запускается CGI-приложение

Как же работает интерфейс? В оставшейся части книги мы будем отвечать на этот вопрос в подробностях, а сейчас рассмотрим основы.

Веб-броузеры запрашивают динамические ресурсы (в том числе CGI-сценарии) так же, как любые другие ресурсы в сети: они посылают сообщение, соответствующее стандартам протокола передачи гипертекста (Hypertext Transfer Protocol, HTTP). HTTP мы обсудим позднее, в главе 2. HTTP-запрос включает универсальный адрес ресурса (Universal Resource Locator, URL), согласно которому веб-сервер определяет, какой ресурс возвращать. Обычно CGI-сценарии находятся в привычном каталоге, например */cgi* и имеют одно и то же расширение, например *.cgi*. Если веб-сервер распознает, что был сделан запрос к CGI-сценарию, то он запускает этот сценарий.

Допустим, вы хотите посетить URL *http://www.mikesmechanics.com/cgi/welcome.cgi*. В примере 1-1 приведен пример HTTP-запроса, который должен послать ваш броузер.

Пример 1-1. HTTP-запрос

```
GET /cgi/welcome.cgi HTTP/1.1
Host: www.mikemechanics.com
```

Этот запрос, имеющий тип «GET», идентифицирует ресурс, который необходимо получить, – */cgi/welcome.cgi*. Предположим, что наш сервер распознает все файлы в каталоге */cgi* как CGI-сценарии, поэтому он запустит сценарий *welcome.cgi* вместо того, чтобы передать его содержимое напрямую в броузер.

CGI-программы получают данные со стандартного ввода (STDIN) и переменных окружения. Эти переменные содержат информацию об удаленном хосте и пользователе, значения элементов отправленной формы (если таковые были) и т. д. Они также содержат имя сервера, протокол и название программного обеспечения, на котором работает сервер. В подробностях мы рассмотрим эти вопросы в главе 3.

Как только CGI-программа запущена, она посылает обработанные данные обратно серверу через стандартный вывод (STDOUT). В Perl это сделать просто, потому как по умолчанию все, что выводится при помощи функции *print*, попадает на стандартный вывод. CGI-сценарии могут возвращать результаты в виде нового документа, либо адреса, на который пересылается запрос. CGI-сценарии содержат специальную строку, отформатированную в соответствии с заголовками HTTP. Эти заголовки мы рассмотрим в следующей главе, а сейчас приведем пример строки, которую должен содержать CGI-сценарий, возвращающий HTML-код:

```
Content-type: text/html
```

CGI-сценарии могут содержать дополнительные строки заголовков, поэтому после заголовка должна быть пустая строка, чтобы отделить сам заголовок от содержимого вывода. Наконец если сценарий возвращает документ, то выводится и содержимое этого документа.

Веб-сервер получает вывод CGI-сценария и добавляет собственные HTTP-заголовки перед его отправкой броузеру пользователя, который запросил сценарий. В примере 1-2 приведен пример ответа, получаемого веб-броузером от веб-сервера.

Пример 1-2. Пример HTTP-ответа

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2000 20:35:35 GMT
Server: Apache/1.3.9 (Unix)
Last-Modified: Wed, 20 May 1998 14:59:42 GMT
ETag: "74916-656-3562efde"
Content-Length: 2000
Content-Type: text/html

<HTML>
```

```
<HEAD>
  <TITLE>Добро пожаловать в базу данных Майка</TITLE>
</HEAD>

<BODY BGCOLOR="#ffffff">
  <IMG SRC="/images/mike.jpg" ALT="Mike's Mechanics">
  <P>Добро пожаловать пришедшему с dyn34.my-isp.net! Что вы тут найдете?
  Вы найдете список оборудования и возможные варианты обслуживания,
  в соответствии с вашими запросами и  предложениями. </P>
  <P>Чего вы ждете? Жмите <A HREF="/cgi/list.cgi">эту ссылку</A>,  что-
бы продолжить.</P>
  <HR>
  <P>Текущее время на сервере: Суббота Март 18 10:28:00 2000.</P>
  <P>Если у вас возникнут проблемы при посещении этого сайта или
  если у вас есть предложения, пожалуйста, напишите веб-мастеру
  <A HREF=mailto:webmaster@mikesmechanics.com>
  webmaster@mikesmechanics.com</A>.</P>
</BODY>
</HTML>
```

Заголовок содержит: протокол, дату и время ответа, имя сервера и версия, дата последнего изменения документа, тег, используемый для кэширования, длину ответа и тип документа в том случае, если документ отформатирован при помощи HTML. Подобные заголовки возвращаются со всеми ответами от веб-сервера; HTTP-заголовки подробно рассмотрены в следующей главе. Обратите внимание, что нет указаний на то, получен ли ответ от статического HTML-файла или создан CGI-сценарием. Именно так и должно быть. Броузер запрашивает у веб-сервера ресурс и получает его. Неважно, как и откуда был получен этот ресурс.

CGI позволяет создавать документ, который ничем не отличается для пользователя от других получаемых по сети документов. Эта гибкость позволяет вам создавать при помощи CGI-сценариев все, что можно получить в виде файла, включая HTML-документы, обычный текст, PDF-файлы и даже изображения в формате PNG или GIF. Как создавать динамические изображения, показано в главе 13.

Пример CGI-сценария

Рассмотрим простое CGI-приложение, написанное на Perl и создающее динамический вывод, который мы видели в примере 1-2. Эта программа (пример 1-3) определяет, откуда пришел пользователь, и затем создает HTML-документ, содержащий эту информацию и текущее время. В нескольких следующих главах показано, как создавать приложения на основе модулей. Пока же мы все сделаем просто.

Пример 1-3. Текст программы welcome.cgi

```
#!/usr/bin/perl -wT

use strict;

my $time        = localtime;
my $remote_id   = $ENV{REMOTE_HOST} || $ENV{REMOTE_ADDR};
my $admin_email = $ENV{SERVER_ADMIN};

print "Content-type: text\html\n\n";

print <<END_OF_PAGE;
<HTML>
<HEAD>
  <TITLE>Добро пожаловать в базу данных Майка</TITLE>
</HEAD>

<BODY BGCOLOR="#ffffff">
  <IMG SRC="/images/mike.jpg" ALT="Mike's Mechanics">
  <P>Добро пожаловать пришедшему с $remote_id! Что вы тут найдете?
  Вы найдете список оборудования и возможные варианты обслуживания,
  в соответствии с вашими запросами и  и предложениями. </P>
  <P>Чего вы ждете?  Жмите <A HREF="/cgi/list.cgi">эту
      ссылку</A>, чтобы продолжить.</P>
  <HR>
  <P>Текущее время на сервере: $time</P>
  <P>Если у вас возникнут проблемы при посещении этого сайта
  или если у вас есть предложения, пожалуйста, напишите веб-
  мастеру <A HREF=mailto:$admin_email>$admin_email</A>.</P>
</BODY>
</HTML>
END_OF_PAGE
```

Эта программа очень проста – она содержит всего 6 команд (последняя состоит из нескольких строк). Посмотрим, как она работает. Поскольку это наш первый сценарий, к тому же очень короткий, мы рассмотрим его строка за строкой (как сказано в предисловии, вы должны иметь представление о Perl).

Первая строка сценария одинакова для большинства сценариев на Perl. Она предписывает серверу использовать программу, находящуюся в */usr/bin/perl*, для запуска и интерпретации сценария. Ключи *-wT* сообщают интерпретатору Perl, что должны быть включены все предупреждения и режим пометки. Предупреждения помогают найти неочевидные проблемы, не вызывающие синтаксических ошибок. Включать предупреждения не обязательно, но очень полезно. Режим пометки стоит использовать во всех сценариях, если только вы не любитель приключений. Подробно мы рассмотрим эту возможность в главе 8.

Команда *use strict* говорит о том, что надо использовать строгие правила для переменных, подпрограмм и ссылок. Если раньше вы никогда не пользовались этой командой, то стоит привыкнуть использовать ее в своих CGI-сценариях. Как и предупреждения, она помогает выявить неочевидные ошибки, например опечатки, которые не вызвали бы синтаксических ошибок. Кроме того, вы приучаетесь к хорошему стилю программирования, объявляя переменные и уменьшая число глобальных переменных. В результате получается код, который гораздо проще сопровождать. Наконец, как вы увидите в главе 17, это просто необходимо при использовании модулей FastCGI и *mod_perl*. Если вы решите пользоваться этими технологиями, начинайте использовать прагму *strict* уже сейчас.

А теперь за дело. Для начала определим три переменные. Переменной $time присваивается значение строки, содержащей текущие дату и время. Переменной `$remote_id` присваивается доменное имя удаленной машины, запрашивающей страницу, полученное из переменных окружения REMOTE_HOST или REMOTE_ADDR. Как упоминалось выше, CGI-сценарии получают всю информацию с веб-сервера из переменных окружения и со стандартного ввода. Переменная REMOTE_HOST содержит полное доменное имя удаленной машины, но только в том случае, если на веб-сервере разрешено обратное разыменование адресов, в противном случае значение этой переменной пусто. В этом случае используется переменная REMOTE_ADDR, которая содержит IP-адрес удаленной машины. Третьей переменной `$admin_email` присваивается значение переменной окружения SERVER_ADMIN, то есть адрес администратора сервера, соответственно конфигурационным файлам сервера. Это только некоторые из переменных окружения, доступных CGI-сценариям. Мы рассмотрим эти и остальные переменные в главе 3.

Как было сказано выше, если CGI-сценарий возвращает новый документ, сначала он должен вернуть HTTP-заголовок, объявляющий тип возвращаемого документа. Помимо этого, добавляется пустая строка, говорящая о том, что посылка заголовков закончена. Затем печатается тело документа.

Вместо того чтобы использовать функцию *print* для отправки каждой отдельной строки на стандартный вывод, мы используем так называемый «here-документ», который позволяет напечатать за один раз целый блок текста. Это стандартная возможность Perl, не самая очевидная. Эта команда предписываетт Perl выводить все последующие строки до тех пор, пока в отдельной строке не встретится `END_OF_PAGE`. Текст воспринимается так, как будто он заключен в двойные кавычки, переменные вычисляются, но кавычки не отображаются на экране. «Here-документ» не только спасает нас от излишнего печатания, но и делает программу удобочитаемой. Для HTML-вывода существуют и другие пути, речь о которых пойдет в главах 5 и 6.

Вот и все с нашим сценарием. Веб-сервер добавляет дополнительные HTTP-заголовки и возвращает ответ клиенту, как показано в примере 1-2. Если у вас остались вопросы или понятны не все детали, мы рассмотрим их ниже.

Вызов CGI-сценариев

У CGI-сценариев есть свой собственный адрес, как и у HTML-страниц и других ресурсов в сети. Обычно веб-сервер настроен так, что определенный виртуальный каталог (каталог, входящий в URL) указывает на наличие в нем CGI-сценариев, например, в таких каталогах, как */cgi-bin*, */cgi*, */scripts* и пр. Обычно и местоположение CGI-сценариев на сервере и соответствующие им веб-адреса можно переопределить в настройках сервера. Как сделать это для веб-сервера Apache, мы расскажем чуть позже в разделе «Настройка CGI-сценариев».

В Unix исполняемые файлы отличаются от других. CGI-сценарии должны быть исполняемыми. Предположим, что у вас есть файл с именем *my_script.cgi*, написанный на Perl. Чтобы сделать этот файл исполняемым, выполните в интерпретаторе команду:

```
chmod 0755 my_script.cgi
```

Очень часто об этом шаге забывают. Возможно, другие операционные системы по-другому настраиваются для выполнения сценариев. Обратитесь к документации по вашему веб-серверу.

Альтернативные технологии

Судя по названию, эта книга посвящена CGI-программам, написанным на Perl. Поскольку Perl и CGI очень часто используются вместе, некоторые люди не всегда знают об их различиях. Perl – это язык программирования, а CGI – интерфейс, который используется программой для обработки запросов от веб-сервера. Есть альтернативы как CGI, так и Perl: несколько альтернатив CGI позволяют обрабатывать динамические запросы, да и CGI-приложения можно писать на любых языках.

Почему Perl?

Хотя CGI-приложения можно писать практически на любом языке, Perl и CGI-программирование стали синонимами для многих программистов. Как сказал Хасан Шрейдер (Hassan Shroeder), первый веб-мастер Sun, «Perl – это артерия Интернета». Perl – самый широко исполь-

зуемый язык для CGI-программирования, и для этого есть много веских причин:

- Perl легко выучить: его синтаксис напоминает другие языки (например C), потому что он «многое прощает», – при ошибке выдается подробное сообщение, помогающее быстро локализовать проблему.
- Perl способствует быстрой разработке, так как это интерпретируемый язык; исходный код не надо компилировать перед запуском.
- Perl доступен на многих платформах с минимальными изменениями.
- Perl содержит очень мощные функции для обработки строк со встроенной в язык поддержкой поиска и замены по регулярным выражениям.
- Perl обрабатывает двоичные данные так же легко, как и текст.
- Perl не требует четкого разделения на типы: числа, строки и логические выражения являются обычными скалярами.
- Perl взаимодействует с внешними приложениями очень просто и обеспечивает собственные функции для работы с файловыми системами.
- Для Perl есть много свободно доступных модулей от CPAN, начиная с модулей для создания динамической графики до интерфейсов с Интернет-серверами и системами управления базами данных. За подробной информацией по CPAN обратитесь к приложению B.

Perl действительно очень быстрый: считывая исходный файл, он тут же компилирует его в низкоуровневый код, который потом исполняет. Обычно компиляция и исполнение в Perl не воспринимаются как отдельные шаги, поскольку выполняются вместе: Perl запускается, читает исходный файл, компилирует его, запускает и затем завершает работу. Этот процесс повторяется каждый раз, когда запускается сценарий Perl, в том числе CGI-сценарии. Поскольку Perl так эффективен, этот процесс происходит достаточно быстро, чтобы обрабатывать все запросы не на самых загруженных серверах. Обратите внимание, что в системах Windows это гораздо менее эффективно из-за необходимости создания новых процессов.

Альтернативы CGI

В последние годы появилось несколько альтернатив CGI. Все они наследуют CGI и обеспечивают собственный подход к одной и той же цели: отвечать на запросы и предоставлять динамическое содержимое через HTTP. Большинство из них пытается избежать основного недостатка CGI-сценариев: создания отдельного процесса для запуска сценария каждый раз, когда он запрашивается. Другие пытаются стирать

различия между HTML-страницами и кодом, помещая код в HTML-страницы. Теории, лежащие в основе этого подхода, мы обсудим в главе 6. Вот основные альтернативы CGI:

ASP

Активные серверные страницы (Active Server Pages, ASP), созданные Microsoft для собственного веб-сервера, сейчас доступны для многих серверов. Сервер ASP интегрирован в веб-сервер и не требует отдельного процесса. Он позволяет программистам совмещать код и HTML-страницы вместо того, чтобы писать отдельные программы. Как будет показано в главе 6, существуют модули, позволяющие делать то же самое, используя CGI. ASP поддерживают различные языки программирования, самый популярный из которых Visual Basic, хотя JavaScript также поддерживается. Кроме того, существует версия Perl от ActiveState, которую можно использовать в Windows с ASP. Помимо этого, существует модуль для Perl, Apache::ASP, который поддерживает ASP с *mod_perl*.

PHP

PHP – это язык программирования, сходный с Perl, интерпретатор которого встроен в веб-сервер. PHP поддерживает встроенный код внутри HTML-страниц. PHP поддерживается веб-сервером Apache.

ColdFusion

ColdFusion от Allaire в большей степени чем PHP различает страницы с кодом и HTML-страницы. В HTML-страницах могут быть дополнительные теги, вызывающие функции ColdFusion. В ColdFusion доступны несколько стандартных функций, и разработчики могут создавать собственные функции как расширения. ColdFusion был первоначально написан для Windows, но теперь доступны версии и для Unix. Интерпретатор ColdFusion встроен в веб-сервер.

Java-сервлеты

Java-сервлеты были созданы в Sun. Сервлеты похожи на CGI-сценарии тем, что это код, создающий документы. Тем не менее, сервлеты, поскольку они используют Java, должны быть скомпилированы перед запуском как классы, которые динамически загружаются веб-сервером при запуске сервлетов. Интерфейс отличается от CGI. JavaServer Pages или JSP – это другая технология, позволяющая разработчикам встраивать Java в веб-страницы, наподобие ASP.

FastCGI

FastCGI поддерживает работу одного или более экземпляров *perl*, которые постоянно запущены вместе с интерфейсом, позволяющим передавать динамические запросы от сервера к ним. Это помогает избежать основного недостатка CGI, когда для каждого запроса создается отдельный процесс. При этом FastCGI остается во многом совместимым с CGI. FastCGI доступен на многих веб-серверах. Подробно FastCGI рассматривается в главе 17.

mod_perl

mod_perl – это модуль для веб-сервера Apache, также позволяющий не запускать отдельный экземпляр *perl* для каждого сценария. Вместо того чтобы поддерживать отдельный экземпляр *perl*, как в FastCGI, *mod_perl* встраивает интерпретатор *perl* в веб-сервер. Это обеспечивает преимущества в производительности, а также позволяет получить доступ из кода на Perl к внутренним данным Apache. Мы обсудим *mod_perl* в главе 17.

Стремительное развитие этих конкурирующих технологий не мешает CGI попрежнему оставаться самым популярным методом для передачи динамических страниц и, несмотря на пророчества, не сойдет «со сцены» в ближайшее время. Даже если вы решите использовать другие технологии, изучить CGI очень полезно. Поскольку CGI – очень прозрачный интерфейс, вы разберетесь, как работают веб-транзакции на самом низком уровне, и это поможет вам понять другие технологии, построенные на той же основе. Кроме того, CGI универсален. Многие технологии требуют, чтобы у вас была установлена особенная комбинация других приложений помимо веб-сервера. CGI поддерживается практически любым веб-сервером без дополнительных действий и будет оставаться таким и в дальнейшем.

Конфигурация веб-сервера

До запуска CGI-программы на вашем сервере необходимо изменить некоторые параметры в конфигурационных файлах сервера. На протяжении всей книги мы будем говорить о веб-сервере Apache на платформе Unix. Apache – самый популярный из доступных серверов, к тому же он свободно распространяется и доступен вместе с исходными кодами. Apache возник из веб-сервера NCSA, поэтому многие элементы настройки похожи на подобные для других веб-серверов, имеющих тот же источник, например продаваемых iPlanet (бывший Netscape).

Кроме того, мы предполагаем, что у вас есть доступ к работающему веб-серверу, поэтому не рассматриваем вопросы установки и конфигурирования Apache. Этому посвящена книга *Apache: The Definitive Guide* («Apache, установка и использование») Бена и Питера Лори (Ben and Peter Laurie), издательство O'Reilly & Associates, Inc.

Apache не всегда установлен в одном и том же месте на разных платформах. На протяжении всей книги мы будем считать, что Apache установлен в пути по умолчанию, то есть все находится в каталоге */usr/local/apache*. Подкаталоги Apache:

```
$ cd /usr/local/apache
$ ls -F
bin/ cgi-bin/ conf/ htdocs/ icons/ include/ libexec/ logs/ man/ proxy/
```

В зависимости от того, как сервер Apache был настроен во время установки, у вас может не быть отдельных каталогов, например *libexec* и *proxy*; это не беда. Иногда в популярных дистрибутивах, включающих Apache (например, в некоторых дистрибутивах Linux), эти подкаталоги могут в системе находиться в другом месте. Например, в дистрибутиве RedHat Linux каталоги располагаются, как показано в таблице 1-1.

Таблица 1-1. Альтернативные пути к важным каталогам Apache

Путь по умолчанию	Альтернативный путь (RedHat Linux)
/usr/local/apache/cgi-bin	*/home/httpd/cgi-bin*
/usr/local/apache/htdocs	*/home/httpd/html*
/usr/local/apache/conf	*/etc/httpd/conf*
/usr/local/apache/logs	*/var/log/httpd*

Если в вашей системе пути иные, вы должны пересматривать все наши инструкции относительно путей соответственно вашему случаю. Если у вас установлен сервер Apache, но его каталогов нет ни в одном из указанных путей, то чтобы их найти, обратитесь к системному администратору или к документации.

Apache настраивается изменением конфигурационных файлов из каталога *conf*. В этих файлах содержатся директивы, которые Apache читает и выполняет. В старых версиях Apache было три файла: *httpd.conf*, *srm.conf* и *access.conf*. В новых версиях сервера все директивы находятся в файле *httpd.conf*, и можно настраивать конфигурацию в одном файле. Это также помогает избежать ситуации, когда параметры конфигурации в разных файлах не совпадают, что может вызвать проблемы в безопасности.

На многих серверах по-прежнему используются все три файла, скажем, потому, что администратору недосуг собрать все вместе. Поэтому обсуждая в книге конфигурацию Apache, мы приведем имя альтернативного файла, который надо отредактировать, если в системе есть все три файла.

Наконец, запомните, что Apache должен перечитать свои конфигурационные файлы, когда вы вносите в них изменения. Не требуется перезапускать сервер, хотя это тоже сработает. Если в вашей системе есть команда *apachectl* (часть стандартной установки), вы заставите Apache перечитать конфигурационные файлы (не перезапуская сервер) при помощи команды:

```
$ apachectl graceful
```

Для этого могут потребоваться права администратора сервера (*root*).

Настройка CGI-сценариев

Разрешить исполнение CGI-сценариев в Apache просто, есть два способа сделать это – хороший и не очень. Начнем с хорошего способа, который требует создания специального каталога для ваших CGI-сценариев.

Настройка по каталогу

Директива `ScriptAlias` предписывает веб-серверу связать виртуальный путь, отображаемый в URL, с каталогом на диске и исполнять все файлы, находящиеся в этом каталоге, если они являются CGI-сценариями.

Чтобы разрешить выполнение CGI-сценариев на своем веб-сервере, добавьте в файл *httpd.conf* директиву:

```
ScriptAlias  /cgi  /usr/local/apache/cgi-bin/
```

Теперь, если пользователь обращается к URL

http://your_host.com/cgi/my_script.cgi

сервер выполняет локальную программу

```
/usr/local/apache/cgi-bin/my_script.cgi
```

Обратите внимание, что путь *cgi* в адресе не обязательно должен совпадать с именем каталога файловой системы *cgi-bin*. Когда вы связываете каталог CGI с виртуальным каталогом *cgi*, *cgi-bin* и т. п., это только ваш выбор. У вас может быть несколько каталогов, содержащих CGI-сценарии:

```
ScriptAlias  /cgi   /usr/local/apache/cgi-bin/
ScriptAlias  /cgi2  /usr/local/apache/alt-cgi-bin/
```

Каталог, в котором расположены CGI-сценарии, должен располагаться за пределами корневого каталога сервера. В стандартной установке Apache корневой каталог привязан к каталогу *htdocs*. Все подкаталоги этого каталога можно просматривать. По умолчанию каталог *cgi-bin* расположен вне каталога *htdocs*, чтобы в случае отключения директивы `ScriptAlias` исходный код CGI-сценариев не был доступен всем пользователям по HTTP. Есть и другая веская причина сделать это, не только для того, чтобы кто-либо случайно не удалил директиву `ScriptAlias`.

Вот пример того, почему нельзя помещать каталог с CGI-сценариями в корневом каталоге сервера. Допустим, вам требуется несколько каталогов для CGI-сценариев в корневом каталоге сервера. Вы можете решить завести для каждого серьезного приложения собственный каталог. Допустим, у вас в каталоге */usr/local/apache/htdocs/widgets* есть база элементов управления, а CGI-сценарии расположены в каталоге /

usr/local/apache/htdocs/widgets/cgi. Затем вы добавляете следующую директиву:

```
ScriptAlias  /widgets-cgi  /usr/local/apache/htdocs/widgets/cgi
```

Если, сделав это, вы протестируете работу, то все будет выглядеть нормально. Но предположим, что ваша компания расширилась и кроме widgets продает woozles, поэтому для базы нужно придумать более общее имя. Вы переименовываете каталог *widgets* в *store*, обновляете директиву `ScriptAlias`, обновляете все ссылки на HTML-файлы и создаете символическую ссылку с каталога *widgets* на *store*, чтобы не потерять пользователей, сделавших у себя закладки на старые каталоги. Звучит здорово, правда?

К сожалению, последний шаг – создание символической ссылки – просто создает громадную прореху в безопасности. Проблема в том, что ваши CGI-сценарии доступны по двум адресам. Например, у вас может быть сценарий с именем *purchase.cgi*, к которому теперь можно обратиться двумя различными способами:

http://localhost/store-cgi/purchase.cgi

http://localhost/widgets-cgi/purchase.cgi

Первый адрес обслуживается директивой `ScriptAlias`, второй – нет. Если пользователи придут по второму адресу, то вместо странички они увидят исходный код вашего CGI-сценария. Если вам повезет, кто-нибудь по электронной почте обратит ваше внимание на эту проблему. А если нет, то злонамеренные пользователи начнут изучать исходные тексты ваших сценариев, чтобы найти дыры в безопасности и проникнуть в вашу систему и заполучить более ценную информацию (например, базу данных паролей или номера кредитных карт).

Любая символическая ссылка над каталогом, содержащим CGI-сценарии, приводит к появлению этой дыры в безопасности.[1] Переименование каталога и создание ссылки на старый – всего лишь один пример того, как неприятная ситуация может возникнуть случайно. Если CGI-сценарии будут находиться за пределами корневого каталога сервера, с подобной проблемой вы никогда не столкнетесь.

Почему доступность исходного кода – проблема? Некоторые характеристики CGI-сценариев делают их отличными от исполняемых файлов других типов с точки зрения безопасности. Они позволяют удаленным, анонимным пользователям запускать программы в вашей системе.

[1] Можно настроить Apache так, чтобы не происходило разыменование символических ссылок. Это будет альтернативное решение. Тем не менее, символические ссылки полезны и разрешены по умолчанию. Проблема в данном случае не в символических ссылках, а в том, что каталог с CGI-сценариями можно просматривать.

Поэтому о безопасности всегда надо помнить, и ваш код должен быть настолько безупречен, что даже если вы захотите сделать его доступным, потенциальные злоумышленники не смогли бы этим воспользоваться. И хотя сама по себе безопасность, достигаемая скрытностью, не очень эффективна, она, без сомнения, не повредит, если используется наряду с другими формами безопасности. Вопросы безопасности подробно рассматриваются в главе 8.

Настройка по расширению

Альтернативой настройке CGI-сценариев через общий каталог является указание веб-серверу распознавать CGI-сценарии по их расширению, например *.cgi*. При этом сами сценарии не должны находиться в одном определенном каталоге, а могут храниться в любом месте. Это откровенно плохая идея, как с точки зрения архитектуры, так и безопасности.

С точки зрения архитектуры, это плохо из-за того, что управлять проще сценариями, которые находятся в одном каталоге. По мере роста веб-сайта будет все сложнее следить за сценариями, используемыми на сайте. В одном каталоге гораздо проще найти нужный сценарий или создать сценарий, являющийся общим решением для разных задач, вместо дюжины сценариев, каждый из которых используется только для одной задачи. В каталоге */cgi* можно создать подкаталоги, чтобы организовать ваши сценарии.

Существует две причины, по которым настройка CGI-сценариев по расширению небезопасна. Во-первых, любой, у кого есть права на обновление HTML-файлов, сможет создавать свои CGI-сценарии. Как уже говорилось, CGI-сценарии требуют особого внимания к безопасности, поэтому не стоит разрешать новичкам в программировании создавать сценарии на серьезных веб-серверах. Во-вторых, увеличивается риск того, что кому-то удастся просмотреть исходный код ваших CGI-сценариев. Многие текстовые редакторы создают резервные копии файлов при редактировании; некоторые из них создают эти копии в текущем каталоге. Например, если вы редактируете файл *top_secret.cgi* при помощи *emacs*, обычно создается резервная копия файла с именем *top_secret.cgi~*. Если этот файл создается на веб-сервере, то любой, кто его запросит, получит в результате его исходный текст, поскольку веб-сервер не распознал расширение.

Конечно, в идеале ваш текстовый редактор должен удалять эти файлы после завершения работы с ними, а вы не должны редактировать файлы прямо на веб-сервере. Но подобные файлы иногда остаются. Кроме того, файлы иногда переименовывают вручную. Разработчик может захотеть внести изменения в файл и сделать резервную копию, сохранив старую версию с расширением *.bak*. Если бы резервная копия была в каталоге, настроенном с помощью ScriptAlias, то этот файл не отобра-

жался бы, а просто воспринимался и выполнялся как другой CGI-сценарий, что гораздо безопаснее.

Если ваш веб-сервер позволяет выполнять сценарии с любого места, то вот как можно это исправить. Следующая строка предписывает веб-серверу выполнять любой файл, заканчивающийся суффиксом *.cgi*:

```
AddHandler  cgi-script   .cgi
```

Можно закомментировать эту строку, предварив ее символом #, как в Perl. Без этой директивы Apache будет воспринимать файлы с расширением *.cgi* как файлы неизвестного типа и возвращать их в соответствии с типом по умолчанию, т. е. как обычный текстовый файл. Так что убедитесь перед удалением этой директивы, что вы убрали все CGI-сценарии за пределы корневого каталога сервера.

Можно запретить выполнение CGI-сценариев из определенных каталогов, отключив параметр ExecCGI. Строка, разрешающая это, выглядит следующим образом:

```
<Directory "/usr/local/apache/htdocs">
    .
    .
    Options Indexes FollowSymLinks ExecCGI
    .
    .
</Directory>
```

Вероятно, над и под директивой `Options` есть другие строки, да и сама директива в вашей системе может быть иной. Если вы удалите ExecCGI, то даже при разрешенной директиве обработки CGI Apache не будет выполнять CGI-сценарии в каталоге, относящемся к директиве `Options` – в данном случае в каталоге */usr/local/apache/htdocs*. Вместо этого пользователи увидят страницу с сообщением «Доступ запрещен» (Permission Denied).

Теперь, когда веб-сервер настроен и есть шанс посмотреть, что может CGI, изучим CGI подробнее. Следующая глава начинается с обзора HTTP, который является протоколом Web и основой CGI.