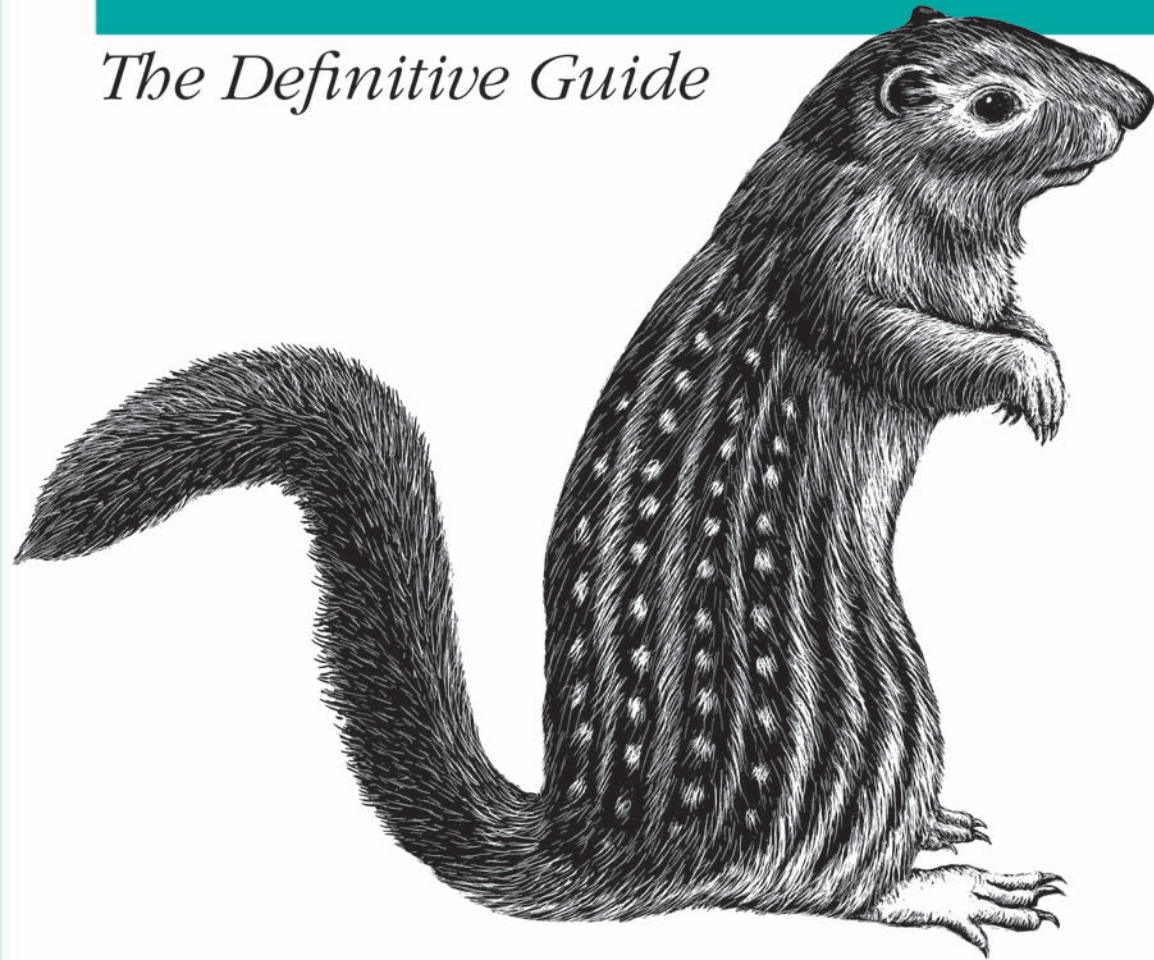


HTTP

The Definitive Guide



O'REILLY®

*David Gourley & Brian Totty
with Marjorie Sayer, Sailu Reddy & Anshu Aggarwal*

HTTP: The Definitive Guide



Behind every successful web transaction lurks the Hypertext Transfer Protocol (HTTP), the language by which web clients and servers exchange documents and information. HTTP is commonly known as the workhorse behind the browsers we use every day to access our company intranets, locate out-of-print books, or research census information.

But HTTP is used for far more than browsing the Web: the simplicity and ubiquity of HTTP also have made it the choice protocol for many other networked applications, most notably through web services such as SOAP and XML-RPC.

As the title suggests, *HTTP: The Definitive Guide* explains the HTTP protocol: how it works and how to use it to develop web-based applications. However, this book is not just about HTTP; it's also about all the other core Internet technologies that HTTP depends on to work effectively. Although HTTP is at the center of the book, the essence of *HTTP: The Definitive Guide* is in understanding how the Web works and how to apply that knowledge to web programming and administration. The book explains the technical workings, motivations, performance considerations, and objectives of HTTP and the technologies around which it revolves.

HTTP: The Definitive Guide is the bible for the HTTP protocol and related web technologies.

Topics covered include:

- HTTP methods, headers, and status codes
- Optimizing proxies and caches
- Strategies for designing web robots and crawlers
- Cookies, authentication, and Secure HTTP
- Internationalization and content negotiation
- Redirection and load-balancing strategies

Written by experts with years of practical experience, this book uses clear, concise language and a plethora of detailed illustrations to help readers visualize what goes on behind the scenes, providing a complete understanding of the story behind each query on the Web.

All web programmers, administrators, and application developers need to be familiar with HTTP in order to work effectively. There are many books that explain how to use the Web, but this is the book that explains how the Web works.

www.oreilly.com

US \$44.95

CAN \$69.95

ISBN: 978-1-56592-509-0



HTTP

The Definitive Guide

HTTP

The Definitive Guide

David Gourley and Brian Totty
with Marjorie Sayer, Sailu Reddy, and Anshu Aggarwal

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

HTTP: The Definitive Guide

by David Gourley and Brian Totty
with Marjorie Sayer, Sailu Reddy, and Anshu Aggarwal

Copyright © 2002 O'Reilly Media, Inc. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol,
CA 95472.

O'Reilly Media, Inc. books may be purchased for educational, business, or sales promotional use. On-line editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Linda Mui
Production Editor: Rachel Wheeler
Cover Designer: Ellie Volckhausen
Interior Designers: David Futato and Melanie Wang

Printing History:

September 2002: First Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. *HTTP: The Definitive Guide*, the image of a thirteen-lined ground squirrel, and related trade dress are trademarks of O'Reilly Media, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN-10: 1-56592-509-2

ISBN-13: 978-1-56592-509-0

[C]

[01/08]

Table of Contents

Preface	xiii
----------------------	-------------

Part I. HTTP: The Web's Foundation

1. Overview of HTTP	3
HTTP: The Internet's Multimedia Courier	3
Web Clients and Servers	4
Resources	4
Transactions	8
Messages	10
Connections	11
Protocol Versions	16
Architectural Components of the Web	17
The End of the Beginning	21
For More Information	21
2. URLs and Resources	23
Navigating the Internet's Resources	24
URL Syntax	26
URL Shortcuts	30
Shady Characters	35
A Sea of Schemes	38
The Future	40
For More Information	41
3. HTTP Messages	43
The Flow of Messages	43
The Parts of a Message	44

Methods	53
Status Codes	59
Headers	67
For More Information	73
4. Connection Management	74
TCP Connections	74
TCP Performance Considerations	80
HTTP Connection Handling	86
Parallel Connections	88
Persistent Connections	90
Pipelined Connections	99
The Mysteries of Connection Close	101
For More Information	104

Part II. HTTP Architecture

5. Web Servers	109
Web Servers Come in All Shapes and Sizes	109
A Minimal Perl Web Server	111
What Real Web Servers Do	113
Step 1: Accepting Client Connections	115
Step 2: Receiving Request Messages	116
Step 3: Processing Requests	120
Step 4: Mapping and Accessing Resources	120
Step 5: Building Responses	125
Step 6: Sending Responses	127
Step 7: Logging	127
For More Information	127
6. Proxies	129
Web Intermediaries	129
Why Use Proxies?	131
Where Do Proxies Go?	137
Client Proxy Settings	141
Tricky Things About Proxy Requests	144
Tracing Messages	150
Proxy Authentication	156

Proxy Interoperation	157
For More Information	160
7. Caching	161
Redundant Data Transfers	161
Bandwidth Bottlenecks	161
Flash Crowds	163
Distance Delays	163
Hits and Misses	164
Cache Topologies	168
Cache Processing Steps	171
Keeping Copies Fresh	175
Controlling Cachability	182
Setting Cache Controls	186
Detailed Algorithms	187
Caches and Advertising	194
For More Information	196
8. Integration Points: Gateways, Tunnels, and Relays	197
Gateways	197
Protocol Gateways	200
Resource Gateways	203
Application Interfaces and Web Services	205
Tunnels	206
Relays	212
For More Information	213
9. Web Robots	215
Crawlers and Crawling	215
Robotic HTTP	225
Misbehaving Robots	228
Excluding Robots	229
Robot Etiquette	239
Search Engines	242
For More Information	246
10. HTTP-NG	247
HTTP's Growing Pains	247
HTTP-NG Activity	248

Modularize and Enhance	248
Distributed Objects	249
Layer 1: Messaging	250
Layer 2: Remote Invocation	250
Layer 3: Web Application	251
WebMUX	251
Binary Wire Protocol	252
Current Status	252
For More Information	253

Part III. Identification, Authorization, and Security

11. Client Identification and Cookies	257
The Personal Touch	257
HTTP Headers	258
Client IP Address	259
User Login	260
Fat URLs	262
Cookies	263
For More Information	276
12. Basic Authentication	277
Authentication	277
Basic Authentication	281
The Security Flaws of Basic Authentication	283
For More Information	285
13. Digest Authentication	286
The Improvements of Digest Authentication	286
Digest Calculations	291
Quality of Protection Enhancements	299
Practical Considerations	300
Security Considerations	303
For More Information	306
14. Secure HTTP	307
Making HTTP Safe	307
Digital Cryptography	309

Symmetric-Key Cryptography	313
Public-Key Cryptography	315
Digital Signatures	317
Digital Certificates	319
HTTPS: The Details	322
A Real HTTPS Client	328
Tunneling Secure Traffic Through Proxies	335
For More Information	336

Part IV. Entities, Encodings, and Internationalization

15. Entities and Encodings	341
Messages Are Crates, Entities Are Cargo	342
Content-Length: The Entity's Size	344
Entity Digests	347
Media Type and Charset	348
Content Encoding	351
Transfer Encoding and Chunked Encoding	354
Time-Varying Instances	359
Validators and Freshness	360
Range Requests	363
Delta Encoding	365
For More Information	369
16. Internationalization	370
HTTP Support for International Content	370
Character Sets and HTTP	371
Multilingual Character Encoding Primer	376
Language Tags and HTTP	384
Internationalized URIs	389
Other Considerations	392
For More Information	392
17. Content Negotiation and Transcoding	395
Content-Negotiation Techniques	395
Client-Driven Negotiation	396
Server-Driven Negotiation	397
Transparent Negotiation	400

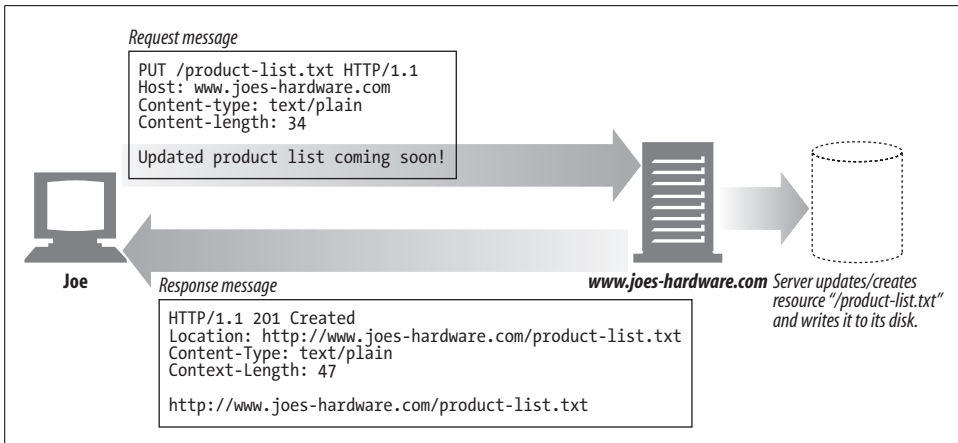
Transcoding	403
Next Steps	405
For More Information	406

Part V. Content Publishing and Distribution

18. Web Hosting	411
Hosting Services	411
Virtual Hosting	413
Making Web Sites Reliable	419
Making Web Sites Fast	422
For More Information	423
19. Publishing Systems	424
FrontPage Server Extensions for Publishing Support	424
WebDAV and Collaborative Authoring	429
For More Information	446
20. Redirection and Load Balancing	448
Why Redirect?	449
Where to Redirect	449
Overview of Redirection Protocols	450
General Redirection Methods	452
Proxy Redirection Methods	462
Cache Redirection Methods	469
Internet Cache Protocol	473
Cache Array Routing Protocol	475
Hyper Text Caching Protocol	478
For More Information	481
21. Logging and Usage Tracking	483
What to Log?	483
Log Formats	484
Hit Metering	492
A Word on Privacy	495
For More Information	495

Part VI. Appendixes

A. URI Schemes	499
B. HTTP Status Codes	505
C. HTTP Header Reference	508
D. MIME Types	533
E. Base-64 Encoding	570
F. Digest Authentication	574
G. Language Tags	581
H. MIME Charset Registry	602
Index	617



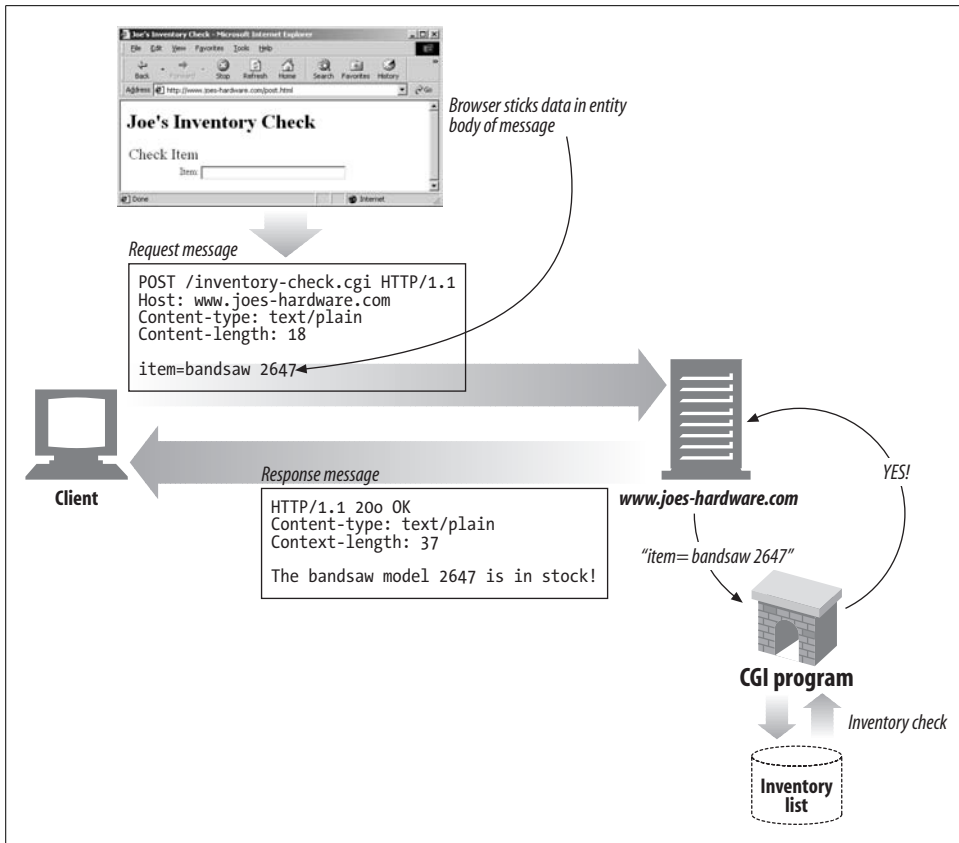


Figure 3-10. POST example

request message it received in the body of its response. A client can then see how, or if, its original message was munged or modified along the request/response chain of any intervening HTTP applications (see Figure 3-11).

The TRACE method is used primarily for diagnostics; i.e., verifying that requests are going through the request/response chain as intended. It's also a good tool for seeing the effects of proxies and other applications on your requests.

As good as TRACE is for diagnostics, it does have the drawback of assuming that intervening applications will treat different types of requests (different methods—GET, HEAD, POST, etc.) the same. Many HTTP applications do different things depending on the method—for example, a proxy might pass a POST request directly to the server but attempt to send a GET request to another HTTP application (such as a web cache). TRACE does not provide a mechanism to distinguish methods. Generally, intervening applications make the call as to how they process a TRACE request.

contains a file named *index.html* (or *index.htm*), the server will return the contents of that file.

In the Apache web server, you can configure the set of filenames that will be interpreted as default directory files using the *DirectoryIndex* configuration directive. The *DirectoryIndex* directive lists all filenames that serve as directory index files, in preferred order. The following configuration line causes Apache to search a directory for any of the listed files in response to a directory URL request:

```
DirectoryIndex index.html index.htm home.html home.htm index.cgi
```

If no default index file is present when a user requests a directory URI, and if directory indexes are not disabled, many web servers automatically return an HTML file listing the files in that directory, and the sizes and modification dates of each file, including URI links to each file. This file listing can be convenient, but it also allows nosy people to find files on a web server that they might not normally find.

You can disable the automatic generation of directory index files with the Apache directive:

```
Options -Indexes
```

Dynamic Content Resource Mapping

Web servers also can map URIs to dynamic resources—that is, to programs that generate content on demand (Figure 5-11). In fact, a whole class of web servers called *application servers* connect web servers to sophisticated backend applications. The web server needs to be able to tell when a resource is a dynamic resource, where the dynamic content generator program is located, and how to run the program. Most web servers provide basic mechanisms to identify and map dynamic resources.

Apache lets you map URI pathname components into executable program directories. When a server receives a request for a URI with an executable path component, it attempts to execute a program in a corresponding server directory. For example, the following Apache configuration directive specifies that all URIs whose paths begin with */cgi-bin/* should execute corresponding programs found in the directory */usr/local/etc/httpd/cgi-programs/*:

```
ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-programs/
```

Apache also lets you mark executable files with a special file extension. This way, executable scripts can be placed in any directory. The following Apache configuration directive specifies that all web resources ending in *.cgi* should be executed:

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

CGI is an early, simple, and popular interface for executing server-side applications. Modern application servers have more powerful and efficient server-side dynamic content support, including Microsoft's Active Server Pages and Java servlets.