

# HTTP 완벽 가이드

웹은 어떻게 동작하는가

# HTTP: THE DEFINITIVE GUIDE

by David Gourley, Brian Totty

with Marjorie Sayer, Sailu Reddy, and Anshu Aggarwal

© Insight Press 2014

Authorized Korean translation of the English edition of HTTP: THE DEFINITIVE GUIDE ISBN 9781565925090 © 2002 O'Reilly Media Inc.

This translation is published and sold by permission of O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

이 책의 한국어판 저작권은 에이전시 원을 통해 저작권자와의 독점 계약으로 인사이트에 있습니다. 신저작권법에 의해 한국 내에서 보호를 받는 저작물이므로 무단전재와 무단복제를 금합니다.

## HTTP 완벽 가이드

**전자책 1쇄 발행** 2022년 5월 10일 (종이책 초판 4쇄 반영) **지은이** 데이빗 고울리, 브라이언 토티, 마조리 세이어, 세일루 레디, 안슈 아가왈 **옮긴이** 이응준, 정상일 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트 **등록번호** 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울특별시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** <http://blog.insightbook.co.kr> **이메일** [insight@insightbook.co.kr](mailto:insight@insightbook.co.kr) **ISBN** 978-89-6626-351-6

프로그래밍인사이트



# HTTP 완벽 가이드

웹은 어떻게 동작하는가

데이빗 고올리, 브라이언 토티, 마조리 셰이어, 세일루 레디, 안슈 아가왈 지음  
이응준, 정상일 옮김

인<인>사이트

# 차례

옮긴이의 글 .....	xviii
서문 .....	xxii

## I. HTTP: 웹의 기초 1

---

### 01. HTTP 개관 3

1.1 HTTP: 인터넷의 멀티미디어 배달부 .....	3
1.2 웹 클라이언트와 서버 .....	4
1.3 리소스 .....	5
1.3.1 미디어 타입 6 / 1.3.2 URI 7 / 1.3.3 URL 8 / 1.3.4 URN 8	
1.4 트랜잭션 .....	9
1.4.1 메서드 9 / 1.4.2 상태 코드 10	
1.4.3 웹페이지는 여러 객체로 이루어질 수 있다 10	
1.5 메시지 .....	11
1.5.1 간단한 메시지의 예 12	
1.6 TCP 커넥션 .....	13
1.6.1 TCP/IP 13 / 1.6.2 접속, IP 주소 그리고 포트번호 14 /	
1.6.3 텔넷(Telnet)을 이용한 실제 예제 15	
1.7 프로토콜 버전 .....	18
1.8 웹의 구성요소 .....	19
1.8.1 프락시 20 / 1.8.2 캐시 20 / 1.8.3 게이트웨이 21 / 1.8.4 터널 21	
1.8.5 에이전트 22	
1.9 시작의 끝 .....	22
1.10 추가 정보 .....	23
1.10.1 HTTP 프로토콜에 대한 정보 23 / 1.10.2 역사적 시각 24	
1.10.3 기타 월드 와이드 웹 정보 25	

### 02. URL과 리소스 27

2.1 인터넷의 리소스 탐색하기 .....	28
2.1.1 URL이 있기 전 암흑의 시대 29	
2.2 URL 문법 .....	30

2,2,1 스킴: 사용할 프로토콜 31 / 2,2,2 호스트와 포트 32	
2,2,3 사용자 이름과 비밀번호 32 / 2,2,4 경로 33 / 2,2,5 파라미터 33	
2,2,6 질의 문자열 34 / 2,2,7 프래그먼트 35	
2.3 단축 URL .....	36
2,3,1 상대 URL 36 / 2,3,2 URL 확장 40	
2.4 안전하지 않은 문자 .....	40
2,4,1 URL 문자 집합 41 / 2,4,2 인코딩 체계 42 / 2,4,3 문자 제한 42	
2,4,4 좀 더 알아보기 43	
2.5 스킴의 바다 .....	44
2.6 미래 .....	46
2,6,1 지금이 아니면, 언제? 46	
2.7 추가 정보 47	
<b>03. HTTP 메시지</b> .....	<b>49</b>
3.1 메시지의 흐름 .....	49
3,1,1 메시지는 원 서버 방향을 인바운드로 하여 송신된다 50	
3,1,2 다운스트림으로 흐르는 메시지 50	
3.2 메시지의 각 부분 .....	50
3,2,1 메시지 문법 52 / 3,2,2 시작줄 54 / 3,2,3 헤더 58 / 3,2,4 엔터티 본문 59	
3,2,5 버전 0,9 메시지 60	
3.3 메서드 .....	60
3,3,1 안전한 메서드(Safe Method) 61 / 3,3,2 GET 61 / 3,3,3 HEAD 62	
3,3,4 PUT 63 / 3,3,5 POST 63 / 3,3,6 TRACE 63 / 3,3,7 OPTIONS 65	
3,3,8 DELETE 66 / 3,3,9 확장 메서드 66	
3.4 상태 코드 .....	67
3,4,1 100-199: 정보성 상태 코드 67 / 3,4,2 200-299: 성공 상태 코드 69	
3,4,3 300-399: 리다이렉션 상태 코드 70 / 3,4,4 400-499: 클라이언트 에러 상태 코드 74	
3,4,5 500-599: 서버 에러 상태 코드 75	
3.5 헤더 .....	76
3,5,1 일반 헤더 77 / 3,5,2 요청 헤더 78 / 3,5,3 응답 헤더 81	
3,5,4 엔터티 헤더 82	
3.6 추가 정보 .....	84
<b>04. 커넥션 관리</b> .....	<b>85</b>
4.1 TCP 커넥션 .....	85
4,1,1 신뢰할 수 있는 데이터 전송 통로인 TCP 86	
4,1,2 TCP 스트림은 세그먼트로 나뉘어 IP 패킷을 통해 전송된다 87	

4.1.3 TCP 커넥션 유지하기 88 / 4.1.4 TCP 소켓 프로그래밍 90	
4.2 TCP의 성능에 대한 고려 .....	91
4.2.1 HTTP 트랜잭션 지연 92 / 4.2.2 성능 관련 중요 요소 93	
4.2.3 TCP 커넥션 핸드셰이크 지연 93 / 4.2.4 확인응답 지연 95	
4.2.5 TCP 느린 시작(slow start) 95 / 4.2.6 네이글(Nagle) 알고리즘과 TCP_NODELAY 96	
4.2.7 TIME_WAIT의 누적과 포트 고갈 97	
4.3 HTTP 커넥션 관리 .....	98
4.3.1 흔히 잘못 이해하는 Connection 헤더 99	
4.3.2 순차적인 트랜잭션 처리에 의한 지연 100	
4.4 병렬 커넥션 .....	101
4.4.1 병렬 커넥션은 페이지를 더 빠르게 내려받는다 102	
4.4.2 병렬 커넥션이 항상 더 빠르지는 않다 102	
4.4.3 병렬 커넥션은 더 빠르게 '느껴질 수' 있다 103	
4.5 지속 커넥션 .....	104
4.5.1 지속 커넥션 vs 병렬 커넥션 104 / 4.5.2 HTTP/1.0+의 Keep-Alive 커넥션 105	
4.5.3 Keep-Alive 동작 106 / 4.5.4 Keep-Alive 옵션 106	
4.5.5 Keep-Alive 커넥션 제한과 규칙 107 / 4.5.6 Keep-Alive와 멍청한(dumb) 프락시 108	
4.5.7 Proxy-Connection 살펴보기 110 / 4.5.8 HTTP/1.1의 지속 커넥션 112	
4.5.9 지속 커넥션의 제한과 규칙 113	
4.6 파이프라인 커넥션 .....	114
4.7 커넥션 끊기에 대한 미스터리 .....	115
4.7.1 '마음대로' 커넥션 끊기 115 / 4.7.2 Content-Length와 Truncation 116	
4.7.3 커넥션 끊기의 허용, 재시도, 멍등성 116 / 4.7.4 우아한 커넥션 끊기 117	
4.8 추가 정보 .....	120
4.8.1 HTTP 커넥션 관련 참고자료 120 / 4.8.2 HTTP 성능 이슈 관련 참고자료 120	
4.8.3 TCP/IP 관련 참고자료 121	

## II. HTTP 아키텍처 123

---

<b>05. 웹 서버</b> .....	<b>125</b>
5.1 다채로운 웹 서버 .....	125
5.1.1 웹 서버 구현 126 / 5.1.2 다목적 소프트웨어 웹 서버 126	
5.1.3 임베디드 웹 서버 127	
5.2 간단한 펄 웹 서버 .....	127
5.3 진짜 웹 서버가 하는 일 .....	130
5.4 단계 1: 클라이언트 커넥션 수락 .....	131

5.4.1 새 커넥션 다루기 131 / 5.4.2 클라이언트 호스트 명 식별 132	
5.4.3 ident를 통해 클라이언트 사용자 알아내기 132	
5.5 단계 2: 요청 메시지 수신 .....	133
5.5.1 메시지의 내부 표현 134 / 5.5.2 커넥션 입력/출력 처리 아키텍처 135	
5.6 단계 3: 요청 처리 .....	137
5.7 단계 4: 리소스의 매핑과 접근 .....	137
5.7.1 Docroot 137 / 5.7.2 디렉터리 목록 140	
5.7.3 동적 콘텐츠 리소스 매핑 141	
5.7.4 서버사이드 인클루드(Server-Side Includes, SSI) 142 / 5.7.5 접근 제어 142	
5.8 단계 5: 응답 만들기 .....	143
5.8.1 응답 엔터티 143 / 5.8.2 MIME 유형 결정하기 143 / 5.8.3 리다이렉션 144	
5.9 단계 6: 응답 보내기 .....	145
5.10 단계 7: 로깅 .....	146
5.11 추가 정보 .....	146
<b>06. 프락시</b> .....	<b>147</b>
6.1 웹 중개자 .....	148
6.1.1 개인 프락시와 공유 프락시 148 / 6.1.2 프락시 대 게이트웨이 149	
6.2 왜 프락시를 사용하는가? .....	150
6.3 프락시는 어디에 있는가? .....	155
6.3.1 프락시 서버 배치 156 / 6.3.2 프락시 계층 157	
6.3.3 어떻게 프락시가 트래픽을 처리하는가 159	
6.4 클라이언트 프락시 설정 .....	161
6.4.1 클라이언트 프락시 설정: 수동 162	
6.4.2 클라이언트 프락시 설정: PAC 파일 162	
6.4.3 클라이언트 프락시 설정: WPAD 163	
6.5 프락시 요청의 미묘한 특징들 .....	164
6.5.1 프락시 URI는 서버 URI와 다르다 164	
6.5.2 가상 호스팅에서 일어나는 같은 문제 165	
6.5.3 인터셉트 프락시는 부분 URI를 받는다 166	
6.5.4 프락시는 프락시 요청과 서버 요청을 모두 다룰 수 있다 167	
6.5.5 전송 중 URI 변경 168	
6.5.6 URI 클라이언트 자동확장과 호스트 명 분석(Hostname Resolution) 168	
6.5.7 프락시 없는 URI 분석(URI Resolution) 169	
6.5.8 명시적인 프락시를 사용할 때의 URI 분석 170	
6.5.9 인터셉트 프락시를 이용한 URI 분석 170	
6.6 메시지 추적 .....	172

6,6,1 Via 헤더 172 / 6,6,2 TRACE 메서드 177	
6.7 프락시 인증 .....	178
6.8 프락시 상호운용성 .....	180
6,8,1 지원하지 않는 헤더와 메서드 다루기 180	
6,8,2 OPTIONS: 어떤 기능을 지원하는지 알아보기 181 / 6,8,3 Allow 헤더 182	
6.9 추가 정보 .....	182
<b>07. 캐시 .....</b>	<b>185</b>
7.1 불필요한 데이터 전송 .....	185
7.2 대역폭 병목 .....	186
7.3 갑작스런 요청 쇄도(Flash Crowds) .....	187
7.4 거리로 인한 지연 .....	188
7.5 적중과 부적중 .....	189
7,5,1 재검사(Revalidation) 190 / 7,5,2 적중률 192 / 7,5,3 바이트 적중률 192	
7,5,4 적중과 부적중의 구별 193	
7.6 캐시 토폴로지 .....	193
7,6,1 개인 전용 캐시 193 / 7,6,2 공용 프락시 캐시 194	
7,6,3 프락시 캐시 계층들 195 / 7,6,4 캐시망, 콘텐츠 라우팅, 피어링 196	
7.7 캐시 처리 단계 .....	198
7,7,1 단계 1: 요청 받기 199 / 7,7,2 단계 2: 파싱 199 / 7,7,3 단계 3: 검색 199	
7,7,4 단계 4: 신선도 검사 199 / 7,7,5 단계 5: 응답 생성 200	
7,7,6 단계 6: 전송 200 / 7,7,7 단계 7: 로깅 200 / 7,7,8 캐시 처리 플로 차트 201	
7.8 사본을 신선하게 유지하기 .....	201
7,8,1 문서 만료 201 / 7,8,2 유효기간과 나이 202 / 7,8,3 서버 재검사 203	
7,8,4 조건부 메서드와의 재검사 204 / 7,8,5 If-Modified-Since: 날짜 재검사 204	
7,8,6 If-None-Match: 엔터티 태그 재검사 206 / 7,8,7 약한 검사기와 강한 검사기 207	
7,8,8 언제 엔터티 태그를 사용하고 언제 Last-Modified 일시를 사용하는가 208	
7.9 캐시 제어 .....	209
7,9,1 no-cache와 no-store 응답 헤더 209 / 7,9,2 Max-Age 응답 헤더 210	
7,9,3 Expires 응답 헤더 210 / 7,9,4 Must-Revalidate 응답 헤더 211	
7,9,5 휴리스틱 만료 211 / 7,9,6 클라이언트 신선도 제약 212 / 7,9,7 주의할 점 213	
7.10 캐시 제어 설정 .....	213
7,10,1 아파치로 HTTP 헤더 제어하기 214	
7,10,2 HTTP-EQUIV를 통한 HTML 캐시 제어 214	
7.11 자세한 알고리즘 .....	216
7,11,1 나이와 신선도 수명 217 / 7,11,2 나이 계산 217	
7,11,3 완전한 나이 계산 알고리즘 220 / 7,11,4 신선도 수명 계산 221	



7.11.5 완전한 서버 신선도 알고리즘 222	
7.12 캐시와 광고 .....	223
7.12.1 광고 회사의 딜레마 224 / 7.12.2 퍼블리셔의 응답 224	
7.12.3 로그 마이그레이션 225 / 7.12.4 적중 측정과 사용량 제한 225	
7.13 추가 정보 .....	226
<b>08. 통합점: 게이트웨이, 터널, 릴레이</b> .....	<b>227</b>
8.1 게이트웨이 .....	228
8.1.1 클라이언트 측 게이트웨이와 서버 측 게이트웨이 229	
8.2 프로토콜 게이트웨이 .....	230
8.2.1 HTTP/*: 서버 측 웹 게이트웨이 231	
8.2.2 HTTP/HTTPS: 서버 측 보안 게이트웨이 232	
8.2.3 HTTPS/HTTP: 클라이언트 측 보안 가속 게이트웨이 233	
8.3 리소스 게이트웨이 .....	233
8.3.1 공용 게이트웨이 인터페이스 235 / 8.3.2 서버 확장 API 236	
8.4 애플리케이션 인터페이스와 웹 서비스 .....	236
8.5 터널 .....	237
8.5.1 CONNECT로 HTTP 터널 커넥션 맺기 238	
8.5.2 데이터 터널링, 시간, 커넥션 관리 240 / 8.5.3 SSL 터널링 241	
8.5.4 SSL 터널링 vs HTTP/HTTPS 게이트웨이 242 / 8.5.5 터널 인증 243	
8.5.6 터널 보안에 대한 고려사항들 243	
8.6 릴레이 .....	244
8.7 추가 정보 .....	246
<b>09. 웹 로봇</b> .....	<b>247</b>
9.1 크롤러와 크롤링 .....	248
9.1.1 어디에서 시작하는가: '루트 집합' 248	
9.1.2 링크 추출과 상대 링크 정상화 249 / 9.1.3 순환 피하기 249	
9.1.4 루프와 중복 250 / 9.1.5 빵 부스러기의 흔적 251	
9.1.6 별칭(alias)과 로봇 순환 252 / 9.1.7 URL 정규화하기 253	
9.1.8 파일 시스템 링크 순환 254 / 9.1.9 동적 가상 웹 공간 255	
9.1.10 루프와 중복 피하기 256	
9.2 로봇의 HTTP .....	259
9.2.1 요청 헤더 식별하기 259 / 9.2.2 가상 호스팅 260	
9.2.3 조건부 요청 261 / 9.2.4 응답 다루기 262 / 9.2.5 User-Agent 타기팅 263	
9.3 부적절하게 동작하는 로봇들 .....	263
9.4 로봇 차단하기 .....	265

9.4.1 로봇 차단 표준 266 / 9.4.2 웹 사이트와 robots.txt 파일들 267  
9.4.3 robots.txt 파일 포맷 268 / 9.4.4 그 외에 알아둘 점 271  
9.4.5 robots.txt의 캐싱과 만료 271 / 9.4.6 로봇 차단 펄 코드 272  
9.4.7 HTML 로봇 제어 META 태그 274

9.5 로봇 에티켓 ..... 277

9.6 검색엔진 ..... 280

9.6.1 넓게 생각하라 280 / 9.6.2 현대적인 검색엔진의 아키텍처 281  
9.6.3 풀 텍스트 색인 281 / 9.6.4 질의 보내기 282  
9.6.5 검색 결과를 정렬하고 보여주기 283 / 9.6.6 스푸핑 284

9.7 추가 정보 ..... 284

**10. HTTP/2.0 ..... 287**

10.1 HTTP/2.0의 등장 배경 ..... 287

10.2 개요 ..... 288

10.3 HTTP/1.1과의 차이점 ..... 289

10.3.1 프레임 289 / 10.3.2 스트림과 멀티플렉싱 290 / 10.3.3 헤더 압축 291  
10.3.4 서버 푸시 292

10.4 알려진 보안 이슈 ..... 293

10.4.1 중개자 캡슐화 공격(Intermediary Encapsulation Attacks) 293  
10.4.2 긴 커넥션 유지로 인한 개인정보 누출 우려 293

10.5 추가 정보 ..... 294

**III. 식별, 인가, 보안 ..... 295**

---

**11. 클라이언트 식별과 쿠키 ..... 297**

11.1 개별 접촉 ..... 297

11.2 HTTP 헤더 ..... 298

11.3 클라이언트 IP 주소 ..... 300

11.4 사용자 로그인 ..... 301

11.5 뚱뚱한 URL ..... 303

11.6 쿠키 ..... 305

11.6.1 쿠키의 타입 305 / 11.6.2 쿠키는 어떻게 동작하는가 305  
11.6.3 쿠키 상자: 클라이언트 측 상태 306 / 11.6.4 사이트마다 각기 다른 쿠키들 308  
11.6.5 쿠키 구성요소 310 / 11.6.6 Version 0(넷스케이프) 쿠키 311  
11.6.7 Version 1 (RFC 2965) 쿠키 312 / 11.6.8 쿠키와 세션 추적 316

11.6.9 쿠키와 캐싱 317 / 11.6.10 쿠키, 보안 그리고 개인정보 319	
11.7 추가 정보 .....	320
<b>12. 기본 인증</b>	<b>321</b>
12.1 인증 .....	322
12.1.1 HTTP의 인증요구/응답 프레임워크 322	
12.1.2 인증 프로토콜과 헤더 323 / 12.1.3 보안 영역 324	
12.2 기본 인증 .....	325
12.2.1 기본 인증의 예 325 / 12.2.2 Base-64 사용자 이름/비밀번호 인코딩 326	
12.2.3 프락시 인증 327	
12.3 기본 인증의 보안 결함 .....	328
12.4 추가 정보 .....	330
<b>13. 다이제스트 인증</b>	<b>331</b>
13.1 다이제스트 인증의 개선점 .....	331
13.1.1 비밀번호를 안전하게 지키기 위해 요약 사용하기 332	
13.1.2 단방향 요약 334 / 13.1.3 재전송 방지를 위한 난스(nonce) 사용 335	
13.1.4 다이제스트 인증 핸드셰이크 335	
13.2 요약 계산 .....	338
13.2.1 요약 알고리즘 입력 데이터 338 / 13.2.2 H(d)와 KD(s,d) 알고리즘 338	
13.2.3 보안 관련 데이터 (A1) 338 / 13.2.4 메시지 관련 데이터 (A2) 339	
13.2.5 요약 알고리즘 전반 340 / 13.2.6 다이제스트 인증 세션 341	
13.2.7 사전(preemptive) 인가 341 / 13.2.8 난스 선택 344 / 13.2.9 상호 인증 345	
13.3 보호 수준(Quality of Protection) 향상 .....	346
13.3.1 메시지 무결성 보호 346 / 13.3.2 다이제스트 인증 헤더 346	
13.4 실제 상황에 대한 고려 .....	347
13.4.1 다중 인증요구 348 / 13.4.2 오류 처리 348	
13.4.3 보호 공간(Protection Space) 349 / 13.4.4 URI 다시 쓰기 349 / 13.4.5 캐시 350	
13.5 보안에 대한 고려사항 .....	350
13.5.1 헤더 부당 변경 350 / 13.5.2 재전송 공격 350	
13.5.3 다중 인증 메커니즘 351 / 13.5.4 사전(dictionary) 공격 351	
13.5.5 악의적인 프락시와 중간자 공격(Man-in-the-Middle Attack) 352	
13.5.6 선택 평문 공격 352 / 13.5.7 비밀번호 저장 353	
13.6 추가 정보 .....	354

<b>14. 보안 HTTP</b>	<b>355</b>
14.1 HTTP를 안전하게 만들기	355
14.1.1 HTTPS	356
14.2 디지털 암호학	357
14.2.1 비밀 코드의 기술과 과학	358 / 14.2.2 암호(cipher) 358
14.2.3 암호 기계	359 / 14.2.4 키가 있는 암호 360 / 14.2.5 디지털 암호 361
14.3 대칭키 암호법	361
14.3.1 키 길이와 열거 공격(Enumeration Attack)	362 / 14.3.2 공유키 발급하기 364
14.4 공개키 암호법	364
14.4.1 RSA	366 / 14.4.2 혼성 암호 체계와 세션 키 366
14.5 디지털 서명	367
14.5.1 서명은 암호 체크섬이다	367
14.6 디지털 인증서	368
14.6.1 인증서의 내부	369 / 14.6.2 X.509 v3 인증서 370
14.6.3 서버 인증을 위해 인증서 사용하기	371
14.7 HTTPS의 세부사항	372
14.7.1 HTTPS 개요	373 / 14.7.2 HTTPS 스킴 373
14.7.3 보안 전송 셋업	374 / 14.7.4 SSL 핸드셰이크 376 / 14.7.5 서버 인증서 377
14.7.6 사이트 인증서 검사	378 / 14.7.7 가상 호스팅과 인증서 379
14.8 진짜 HTTPS 클라이언트	379
14.8.1 OpenSSL	379 / 14.8.2 간단한 HTTPS 클라이언트 381
14.8.3 우리의 단순한 OpenSSL 클라이언트 실행하기	384
14.9 프락시를 통한 보안 트래픽 터널링	386
14.10 추가 정보	388
14.10.1 HTTP 보안	388 / 14.10.2 SSL과 TLS 388
14.10.3 공개키 인프라	389 / 14.10.4 디지털 암호 390

---

## **IV. 엔터티, 인코딩, 국제화** **391**

<b>15. 엔터티와 인코딩</b>	<b>393</b>
15.1 메시지는 컨테이너, 엔터티는 화물	394
15.1.1 엔터티 본문	396
15.2 Content-Length: 엔터티의 길이	397
15.2.1 잘림 검출	397 / 15.2.2 잘못된 Content-Length 398
15.2.3 Content-Length와 지속 커넥션(Persistent Connection)	398
15.2.4 콘텐츠 인코딩	399 / 15.2.5 엔터티 본문 길이 판별을 위한 규칙 399

15.3 엔터티 요약 .....	401
15.4 미디어 타입과 차셋(Charset) .....	402
15.4.1 텍스트 매체를 위한 문자 인코딩 403 / 15.4.2 멀티파트 미디어 타입 403	
15.4.3 멀티파트 폼 제출 403 / 15.4.4 멀티파트 범위 응답 404	
15.5 콘텐츠 인코딩 .....	405
15.5.1 콘텐츠 인코딩 과정 405 / 15.5.2 콘텐츠 인코딩 유형 407	
15.5.3 Accept-Encoding 헤더 407	
15.6 전송 인코딩과 청크 인코딩 .....	408
15.6.1 안전한 전송 409 / 15.6.2 Transfer-Encoding 헤더 410	
15.6.3 청크 인코딩 411 / 15.6.4 콘텐츠와 전송 인코딩의 조합 413	
15.6.5 전송 인코딩 규칙 414	
15.7 시간에 따라 바뀌는 인스턴스 .....	414
15.8 검사기와 신선도 .....	415
15.8.1 신선도 415 / 15.8.2 조건부 요청과 검사기 417	
15.9 범위 요청 .....	419
15.10 델타 인코딩 .....	421
15.10.1 인스턴스 조작, 델타 생성기 그리고 델타 적용기 424	
15.11 추가 정보 .....	425
<b>16. 국제화 .....</b>	<b>427</b>
16.1 국제적인 콘텐츠를 다루기 위해 필요한 HTTP 지원 .....	428
16.2 문자집합과 HTTP .....	428
16.2.1 차셋(Charset)은 글자를 비트로 변환하는 인코딩이다 429	
16.2.2 문자집합과 인코딩은 어떻게 동작하는가 430	
16.2.3 잘못된 차셋은 잘못된 글자들을 낳는다 431	
16.2.4 표준화된 MIME 차셋 값 431	
16.2.5 Content-Type charset 헤더와 META 태그 433	
16.2.6 Accept-Charset 헤더 433	
16.3 다중언어 문자 인코딩에 대한 지침 .....	434
16.3.1 문자집합 용어 434 / 16.3.2 '차셋(Charset)'은 형편없는 이름이다 435	
16.3.3 문자 436 / 16.3.4 글리프(glyphs), 연자(ligatures) 그리고 표현 형태 437	
16.3.5 코딩된 문자집합(Coded Character Set) 438 / 16.3.6 문자 인코딩 구조 440	
16.4 언어 태그와 HTTP .....	444
16.4.1 Content-Language 헤더 445 / 16.4.2 Accept-Language 헤더 446	
16.4.3 언어 태그의 종류 446 / 16.4.4 서브태그 447	
16.4.5 대소문자의 구분 및 표현 447 / 16.4.6 IANA 언어 태그 등록 447	
16.4.7 첫 번째 서브태그: 이름공간 448 / 16.4.8 두 번째 서브태그: 이름공간 449	

16.4.9 나머지 서브태그: 이름공간 450 / 16.4.10 선호 언어 설정하기 450	
16.4.11 언어 태그 참조표 450	
16.5 국제화된 URI	450
16.5.1 국제적 가독성 vs 의미 있는 문자들 451	
16.5.2 URI에서 사용될 수 있는 문자들 451	
16.5.3 이스케이핑과 역이스케이핑(unescaping) 452	
16.5.4 국제 문자들을 이스케이핑하기 453 / 16.5.5 URI에서의 모달 전환 453	
16.6 기타 고려사항	454
16.6.1 헤더와 명세에 맞지 않는 데이터 454 / 16.6.2 날짜 454	
16.6.3 도메인 이름 454	
16.7 추가 정보	455
16.7.1 부록 455 / 16.7.2 인터넷 국제화 455 / 16.7.3 국제 표준 456	
<b>17. 내용 협상과 트랜스코딩</b>	<b>457</b>
17.1 내용 협상 기법	458
17.2 클라이언트 주도 협상	458
17.3 서버 주도 협상	459
17.3.1 내용 협상 헤더 460 / 17.3.2 내용 협상 헤더의 품질값 461	
17.3.3 그 외의 헤더들에 의해 결정 461 / 17.3.4 아파치의 내용 협상 462	
17.3.5 서버 측 확장 463	
17.4 투명 협상	463
17.4.1 캐시와 얼터네이트(alternate) 464 / 17.4.2 Vary 헤더 465	
17.5 트랜스코딩	467
17.5.1 포맷 변환 468 / 17.5.2 정보 합성 468 / 17.5.3 콘텐츠 주입 469	
17.5.4 트랜스코딩 vs. 정적으로 미리 생성해놓기 469	
17.6 다음 단계	470
17.7 추가 정보	471
<b>V. 콘텐츠 발행 및 배포</b>	<b>473</b>
<b>18. 웹 호스팅</b>	<b>475</b>
18.1 호스팅 서비스	476
18.1.1 간단한 예 : 전용 호스팅 476	
18.2 가상 호스팅	477
18.2.1 호스트 정보가 없는 가상 서버 요청 478	

18,2,2 가상 호스팅 동작하게 하기 479 / 18,2,3 HTTP/1,1 Host 헤더 484	
18,3 안정적인 웹 사이트 만들기 .....	486
18,3,1 미러링 된 서버 팜 486 / 18,3,2 콘텐츠 분산 네트워크 487	
18,3,3 CDN의 대리 캐시 488 / 18,3,4 CDN의 프락시 캐시 488	
18,4 웹 사이트 빠르게 만들기 .....	489
18,5 추가 정보 .....	489
<b>19. 배포 시스템</b> .....	<b>491</b>
19.1 배포 지원을 위한 FrontPage 서버 확장 .....	491
19,1,1 FrontPage 서버 확장 492 / 19,1,2 FrontPage 용어 492	
19,1,3 FrontPage RPC 프로토콜 493 / 19,1,4 FrontPage 보안 모델 496	
19,2 WebDAV와 공동 저작 .....	497
19,2,1 WebDAV 메서드 497 / 19,2,2 WebDAV와 XML 498	
19,2,3 WebDAV 헤더 499 / 19,2,4 WebDAV 잠금과 덮어쓰기 방지 501	
19,2,5 LOCK 메서드 502 / 19,2,6 UNLOCK 메서드 505	
19,2,7 속성과 META 데이터 506 / 19,2,8 PROPFIND 메서드 506	
19,2,9 PROPPATCH 메서드 508 / 19,2,10 컬렉션과 이름공간 관리 510	
19,2,11 MKCOL 메서드 510 / 19,2,12 DELETE 메서드 512	
19,2,13 COPY와 MOVE 메서드 513 / 19,2,14 향상된 HTTP/1,1 메서드 516	
19,2,15 WebDAV의 버전 관리 517 / 19,2,16 WebDAV의 미래 518	
19,3 추가 정보 .....	518
<b>20. 리다이렉션과 부하 균형</b> .....	<b>521</b>
20.1 왜 리다이렉션인가? .....	522
20.2 리다이렉션 할 곳 .....	522
20.3 리다이렉션 프로토콜의 개요 .....	523
20.4 일반적인 리다이렉션 방법 .....	526
20,4,1 HTTP 리다이렉션 526 / 20,4,2 DNS 리다이렉션 528	
20,4,3 임의 캐스트 어드레싱 533 / 20,4,4 아이피 맥 포워딩 534	
20,4,5 아이피 주소 포워딩 535 / 20,4,6 네트워크 구성요소 제어 프로토콜 537	
20,5 프락시 리다이렉션 방법 .....	538
20,5,1 명시적 브라우저 설정 538 / 20,5,2 프락시 자동 설정 539	
20,5,3 웹 프락시 자동발견 프로토콜(Web Proxy Autodiscovery Protocol) 541	
20,6 캐시 리다이렉션 방법 .....	547
20,6,1 WCCP 리다이렉션 547	
20,7 인터넷 캐시 프로토콜 .....	551
20,8 캐시 배열 라우팅 프로토콜 .....	553

20.9 하이퍼텍스트 캐싱 프로토콜 .....	557
20.9.1 HTCP 인증 559 / 20.9.2 캐싱 정책 설정 560	
20.10 추가 정보 .....	560
<b>21. 로깅과 사용 추적 .....</b>	<b>563</b>
21.1 로그란 무엇인가? .....	563
21.2 로그 포맷 .....	564
21.2.1 일반 로그 포맷(Common Log Format) 565	
21.2.2 혼합 로그 포맷(Combined Log Format) 566	
21.2.3 넷스케이프 확장 로그 포맷 567 / 21.2.4 넷스케이프 확장 2 로그 포맷 568	
21.2.5 스쿼드(Squid) 프락시 로그 포맷 570	
21.3 적중 계량하기 .....	573
21.3.1 개요 574 / 21.3.2 Meter 헤더 574	
21.4 개인 정보 보호에 대해 .....	575
21.5 추가 정보 .....	577
<b>VI. 부록 .....</b>	<b>579</b>
<hr/>	
<b>부록 A URI 스킴 .....</b>	<b>581</b>
<b>부록 B HTTP 상태 코드 .....</b>	<b>587</b>
B.1 상태 코드 분류 .....	587
B.2 상태 코드 .....	587
<b>부록 C HTTP 헤더 레퍼런스 .....</b>	<b>591</b>
<b>부록 D MIME 타입 .....</b>	<b>619</b>
D.1 배경 .....	620
D.2 MIME 타입 구조 .....	621
D.2.1 분리형 621 / D.2.2 혼합형 621 / D.2.3 멀티파트형 621 / D.2.4 문법 622	
D.3 MIME 타입 IANA 등록 .....	623
D.3.1 등록 트리 623 / D.3.2 등록 절차 624 / D.3.3 등록 규칙 625	
D.3.4 등록 견본 625 / D.3.5 MIME 미디어 타입 등록 626	
D.4 미디어 타입 표 .....	626



D.4.1 application/* 627 / D.4.2 audio/* 646 / D.4.3 chemical/* 648	
D.4.4 image/* 650 / D.4.5 message/* 652 / D.4.6 model/* 653	
D.4.7 multipart/* 654 / D.4.8 text/* 655 / D.4.9 video/* 658 / D.4.10 실험적 타입 659	

<b>부록 E base-64 인코딩</b>	<b>661</b>
E.1 Base-64 인코딩은 이진 데이터를 안전하게 만들어준다	661
E.2 8비트를 6비트로	662
E.3 Base-64 패딩	663
E.4 필 구현	664
E.5 추가 정보	664
<b>부록 F 다이제스트 인증</b>	<b>665</b>
F.1 다이제스트 WWW-Authenticate 지시자들	665
F.2 다이제스트 Authorization 지시자들	667
F.3 다이제스트 Authentication-Info 지시자들	668
F.4 참조 코드	669
F.4.1 파일 “digcalc.h” 669 / F.4.2 파일 “digcalc.c” 670 / F.4.3 파일 “digtest.c” 672	
<b>부록 G 언어 태그</b>	<b>673</b>
G.1 첫 번째 서브태그 규칙	673
G.2 두 번째 서브태그 규칙	674
G.3 IANA에 등록된 언어 태그들	674
G.4 ISO 639 언어 코드	675
G.5 ISO 3166 국가 코드	687
G.6 언어 관리 단체	694
<b>부록 H MIME 문자집합 등록</b>	<b>697</b>
H.1 MIME 문자집합 등기	697
H.2 선호 MIME 이름	698
H.3 등록된 문자집합	698
찾아보기	715

## 옮긴이의 글

---

왜 옮긴이들은 이 책을 번역했는가?

2010년, 같은 회사에 다니고 있었던 옮긴이들은 또 다른 세 명의 개발자와 함께 이 책의 원서를 교재로 삼아 HTTP 스터디를 했다. 당시 스터디를 하면서 웹 프로그래머가 HTTP를 이해하는 게 매우 중요한 일임에도, 이 책은 왜 번역서가 발간되지 않았는지 의아하게 생각했다. 그리고 2년이 지난 2012년, 인사이트에서 이 책을 번역할 사람을 찾는다는 이야기를 듣게 되었고, 바로 번역 프로젝트에 합류했다.

왜 HTTP를 이해하는 것이 중요한가?

월드 와이드 웹을 지탱하는 가장 중요한 기술 두 가지는 HTML과 HTTP이다. 이 두 기술은 팀 버너스-리가 웹을 발명할 때 함께 만들어졌다. 이 둘 중 하나라도 빠지면 웹은 성립하지 않으며, 한편으로 이 둘만 있어도 어떻게든 웹은 성립할 수 있다. 그 중 HTTP는 웹의 구성요소들이 서로 대화를 할 때 사용하는 프로토콜이다. 따라서 HTTP를 이해한다는 것은 웹이 어떻게 동작하는지 이해한다는 것이며, 이를 깊이 이해하면 웹 프로그래밍을 하면서, 웹 서버를 조작하면서, 그리고 네트워크를 관리하면서 정확한 근거에 기반한 올바른 기술적 판단을 내려야 할 때 큰 도움이 된다.

HTTP는 지난 25년간 사용되어 왔으며 앞으로도 계속해서 사용될 것이다. 업계의 유행에 따라 선호하는 언어가 바뀌고 대세인 프레임워크가 바뀌더라도 웹 애플리케이션이 HTTP를 사용한다는 사실은 변하지 않는다. 웹 서비스를 위한 인프라 역시 끊임없이 진보해왔지만 서버, 캐시, 프락시 등 모든 웹의 구성요소들이 HTTP를 사용해서 대화한다는 사실은 변하지 않는다. 따라서 서버사이드 웹 프로그래머, 프론트엔드 웹 프로그래머, 시스템 엔지니어 등 웹과 관련된 일을 하는 이라면 누구라도 HTTP를 공부하는데 시간과 노력을 들이는 게 효과적이고도 안전한 투자가 될 것이다.

이 책의 원서가 발간된 이후 HTTP 명세에도 몇 가지 변화가 있었다. 그도 그럴

것이 원서가 발간된 건 2002년이고, 2014년에 와서야 번역본이 나오게 되었다. 자 그마치 12년이 지났으니, 아무 일도 일어나지 않았다면 오히려 그게 이상한 일일 것이다.

우선 HTTP/1.1이 15년 만에 개정되었다. 이 책의 원서가 쓰인 시점의 최신 HTTP/1.1 명세는 1999년에 발행된 RFC 2616이었으나, 이 책을 번역하는 도중인 2014년 6월에 발행된 RFC 7230, 7231, 7232, 7233, 7234, 7235로 대체되었다. 개정된 명세는 기존 RFC 2616의 오류나 모호한 점들을 7년에 걸쳐 고쳐 작성한 더욱 올바른 HTTP/1.1 명세다.

HTTP/1.1의 정식 후계자가 될 HTTP/2도 조만간 완성된다. 이 책의 번역을 시작했던 2012년부터 HTTP 작업그룹은 HTTP/2의 설계에 착수했으며 차질 없이 계획대로 진행된다면 2015년 2월에 명세가 RFC로 등록될 것이다.

HTTP/2는 기존의 HTTP/1.1의 메시지 구조와 전송 방법을 새로이 고쳐 성능 문제를 획기적으로 개선한 새로운 HTTP이다. 널리 쓰이는 모든 웹브라우저, 즉 구글 크롬, 모질라 파이어폭스, MS 인터넷 익스플로러가 HTTP/2를 지원하고 있거나 곧 지원할 예정이다. HTTP/2는 성능 면에서의 이점이 뚜렷하기 때문에 트위터 같은 몇몇 웹 서비스에 적용되어 이미 사용되고 있으며, 앞으로 웹브라우저들에 탑재됨에 따라 많은 사업자가 자신의 웹 서비스에 HTTP/2를 적용하게 될 것이다.

웁킨이들은 HTTP/2의 중요성을 인지하고 있기에, HTTP-NG를 다룬 원서 10장을 HTTP/2를 소개하는 글을 작성하여 대체하였다. HTTP/2에 대해 자세히 설명하는 것은 이 책의 범위를 한참 넘어서기에 세세히 다루지는 못하였지만, 독자들이 HTTP/2의 개략적인 개념을 이해하는 데에는 충분할 것이다.

이러한 HTTP 명세의 변화가 이 책의 내용을 무의미하게 만드는 것은 아닌지 염려할 수도 있겠지만, 다행히도 그렇지 않다. 우선 HTTP/1.1 개정판은 여전히 HTTP/1.1이다. 버전이 변하지 않았다는 것은 프로토콜 자체에 변화가 없음을 의미한다. 만약 버전이 변하지 않았음에도 프로토콜에 변화가 생긴다면 상호운용성이 커다란 문제가 생기기 때문에 HTTP/1.1 개정판의 저자들은 기존 명세와의 호환성을 유지하기 위해 만전의 노력을 다했다. HTTP/2 역시 메시지 구조는 바뀌지만 메시지에 의미를 부여하는 방법은 별로 변한 것이 없다. 쉽게 말해, HTTP/1.1에서 정의된 헤더들은 HTTP/2에서도 거의 대부분 같은 의미로 쓰이고 있다. 따라서 이 책의 상당 부분이 HTTP/2에서도 여전히 유효하다.

오히려 웁킨이들이 더 신경 써야 했던 것들은 HTTP 명세의 변화와 직접 관계가 없는 부분들이었다. 원서에서는 웹브라우저를 통해 사례를 설명할 때 인터넷 익스

플로러 5와 넷스케이프 네비게이터를 기준으로 삼았으나, 역서에서는 그러한 사례들을 인터넷 익스플로러 10과 구글 크롬 38을 기준으로 고쳐 썼다. 또한 필요에 따라 새로운 내용을 추가로 작성하기도 했다. 예를 들어 국제화를 다룬 14장에는 한국어 사용자들을 위해 euc-kr에 대한 절을 추가하였다. 그 외에도 옴긴이들은 원서에 아직까지도 남아있었던 여러 오류들을 바로잡아 고쳐 썼다.

마지막으로 이 책을 번역하는 과정에서 큰 도움을 주신 많은 분께 감사 말씀을 드리고자 한다. 우선 도서출판 인사이트의 한기성 대표님께 깊이 감사드린다. 이처럼 좋은 책을 번역할 기회를 주셨고, 번역 작업 후반에는 편집자 역할까지 맡아서 꼼꼼하게 원고를 살피고 고쳐주셨다. 초반 편집을 진행해주신 김승호 에디터님과 마무리를 함께 해주신 조은별 님께도 감사드린다.

이 책의 마무리 단계에서 많은 분이 자원해서 원고를 읽어주시고 의견을 주셨다. 강주희, 김군우, 김유승, 김장환, 김재현, 김종진, 김주희, 김필우, 박선욱, 박창우, 변상필, 변용훈, 서승호, 손권남, 엄익훈, 윤영광, 윤정부, 이동현, 이창신, 이항희, 임승진, 정민우, 정성범, 정인수, 정원화, 한규호 님께 감사드린다. 옴긴이들은 이분들의 소중한 의견을 통해 다양한 오자에서부터 치명적인 오역까지 수많은 잘못들을 바로잡을 수 있었다.

HTTP를 이해하고자 하는 모든 이들에게 이 책이 도움이 되길 진심으로 희망한다.



NHN(현 네이버)에 신입사원으로 입사해서 PHP로 코딩하는 웹 프로그래머로 일하고 있을 때, 나는 웹이 어떻게 동작하는지조차 모르고 있었다. 몰라서 답답했지만 뭘 어떻게 공부해야 하는지도 몰랐다. 1년 넘게 헤매고서야 HTTP를 공부해야 한다는 것을 알고 이 책의 원서를 샀다.

스터디 그룹을 통해 이 책을 함께 공부하고, 각종 HTTP 관련 명세들을 읽고, 때때로 HTTP 메일링 리스트에서 질문도 하고, 가끔은 HTTP 관련된 주제로 발표를 하기도 하다가, 드디어 이 책의 번역을 마무리하기에 이르렀다. 덕분에 HTTP를 이해할 수 있게 되었고, 웹에 대해서도 더 잘 알게 되었다. 3년 넘게 HTTP를 공부한

웹이지만 웹 프로그래머로서 충분히 가치 있는 시간이었다.

이제 번역도 끝났으니, 배운 지식을 써먹을 곳을 찾아보아야겠다. 오픈 소스 소프트웨어 프로젝트에 참여한다든가 HTTP 관련 명세에 기여한다든가 써먹을 만한 곳은 무궁무진 할 것 같다.

— 이응준

수많은 기술이 몇 년 만 지나도 묵은 냄새가 날 정도로 빠르게 변화하는 세상에 살고 있다. 반면 큰 변화 없던 HTTP가, WWW의 판이 데스크톱에서 스마트폰으로 넘어가는 과도기에도 여전히 핵심적인 역할을 하고 있는 것을 보면서 기반 기술의 중요성을 실감한다.

넷스케이프라는 브라우저를 통해 처음 WWW를 접하고 십수 년 지난 후에야 개발자가 되어 REST와 함께 HTTP에 대해 알게 되었다. 그러면서 기반 기술에 대한 정리된 지식을 얻고자 했었다. 그렇게 2010년 NHN(현 네이버)에서 만들었던 『HTTP: The Definitive Guide』 원서 스터디 모임에서부터 지금까지 역서를 준비했던 시간을 되돌아보니 뿌듯하다.

마지막으로 부디 이 번역본이 그동안 오픈 소스 및 커뮤니티에서 받은 도움에 대한 자그마한 보답이라도 되었으면 좋겠다.

— 정상일

2014년 11월

## 서문

---

하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol, HTTP)은 월드 와이드 웹(World Wide Web, WWW)에서 통신하는 데 사용하는 프로토콜 프로그램이다. HTTP를 사용하는 방법에는 여러 가지가 있지만, 웹브라우저와 웹 서버 사이에서의 쌍방향 통신에 사용하는 것이 가장 대표적이다.

초기 HTTP는 단순한 프로토콜이기 때문에, 그다지 자세히 이야기할 게 없다고 생각할지도 모르겠다. 하지만 꽤 무거운 이 책을 손에 들고 있지 않은가. 그 단순한 프로토콜인 HTTP에 대해서 어떻게 수백 쪽에 달하는 책을 썼을지 궁금하다면, 이 책의 목차를 보기 바란다. 이 책은 단순히 HTTP 헤더에 대한 참고서가 아니다. 이 책은 진정한 웹 아키텍처 바이블이다.

우리는 HTTP와 밀접하지만 종종 오해받고 있는 규칙들에 대해 알아볼 것이고, 주제 기반으로 HTTP의 모든 내용을 다룬다. 책 전반에 걸쳐서, “HTTP를 어떻게 사용하는가”에 대한 내용뿐만 아니라 “HTTP를 사용하는 이유”에 대해서도 자세히 설명한다. 그리고 독자들이 관련 문서들을 찾아보는 시간을 절약할 수 있도록, HTTP 애플리케이션이 동작하는 데 필요한 HTTP 외의 주요 기술들을 설명한다. 헤더에 대한 참고자료(거의 모든 HTTP 콘텐츠의 기초가 되는 형식들)는, 찾아보기 좋게 알파벳 순서로 작성하여 부록으로 첨부하였으니 참고하기 바란다. 이 내용들이 HTTP를 좀 더 쉽게 사용하는 데 도움이 되기를 바란다.

이 책은 HTTP나 웹의 기본적인 구조를 이해하고 싶어하는 독자를 대상으로 한다. 소프트웨어나 하드웨어 기술자는, HTTP 및 관련 웹 기술에 대한 참고서로 이 책을 활용할 수 있다. 시스템 아키텍트나 네트워크 관리자는 복잡한 웹 아키텍처를 어떻게 설계, 배포, 관리할지에 대한 내용을 이 책에서 배울 수 있다. 성능 기술자 및 분석 전문가는 캐싱과 성능 최적화를 다루는 절들이 도움이 될 것이다. 홍보 전문가나 컨설팅 전문가는 전반적인 웹 기술에 대한 내용을 좀 더 잘 이해하는데 이 책에 있는 개념 중심의 내용들이 도움이 될 것이다.

이 책은 흔히 생기는 오해와 도움이 될 만한 기술을 설명하고, 간편한 참고자료

들을 제공하며, 딱딱하고 읽기 어려운 표준 명세를 이해하기 쉽게 간략히 설명한다. 이 책 한 권으로, 웹을 더 잘 이해하는 데 필요한 필수 기술 및 관련 기술에 대해서 자세히 살펴볼 수 있을 것이다.

이 책은 열정적으로 인터넷 기술을 공유하는 수많은 사람이 엄청난 양의 작업을 함께한 끝에 만들어낸 결실이다. 이 책이 독자들에게 유용하게 쓰이기 바란다.

## 책에서 사용하는 예: 조의 컴퓨터 가게

이 책에서는 기술적인 개념을 설명하기 위해서, “조의 컴퓨터 가게”라는 가상의 온라인 하드웨어 및 가전제품 가게를 예로 사용한다. 이 책에 있는 예들을 테스트해 볼 수 있도록, 실제 조의 컴퓨터 가게의 온라인 상점(<http://www.joes-hardware.com>)을 만들어 놓았다. 우리는 이 책이 서점에 꽂혀 있는 동안에는 이 웹 사이트를 계속 유지하려고 한다.

## 장별 가이드

이 책은 크게 5부로 나뉘는 21개의 장과, 참고자료 및 관련 기술에 대한 조사들로 이루어진 유용한 부록 8개를 제공한다.

- 1부. HTTP: 웹의 기초
- 2부. HTTP 아키텍처
- 3부. 식별, 인가, 보안
- 4부. 엔터티, 인코딩, 국제화
- 5부. 콘텐츠 발행 및 배포
- 6부. 부록

1부 <HTTP: 웹의 기초>에서는 4개 장에 걸쳐 HTTP의 핵심 기술과 웹의 기초에 대해 다룬다.

- 1장 HTTP 개관: HTTP에 대해 개략적으로 살펴본다.
- 2장 URL과 리소스: 통합 자원 지시자(Uniform Resource Locator, URL)의 포맷과 인터넷상에서 URL이 가리키는 리소스의 다양한 형식에 대해 상세히 다룬다. 그리고 URL에서 더 진화한 지시자인 URN에 대한 내용도 다룬다.

- 3장 HTTP 메시지: HTTP 메시지가 어떻게 웹 콘텐츠를 전송하는지 알아본다.
- 4장 커넥션 관리: HTTP 커넥션 관리에 대한 흔한 오해와 잘못 작성된 규칙 및 동작들을 설명한다.

2부 <HTTP의 구조>에서는 HTTP 서버, 프락시, 캐시, 게이트웨이, 로봇 애플리케이션 같은 웹 시스템을 구성하는 빌딩 블록을 주로 다룬다. (물론 웹브라우저 역시 또 하나의 빌딩 블록이지만, 그것에 대해서는 1부에서 이미 다루었다.) 2부는 다음과 같은 6개의 장을 포함한다.

- 5장 웹 서버: 웹 서버 아키텍처에 대한 개요를 제공한다.
- 6장 프락시: HTTP를 전달하고 제어함으로써 플랫폼 역할을 하는 중개 서버인 HTTP 프락시 서버에 대해 알아본다.
- 7장 캐싱: 자주 사용하는 문서를 로컬에 복제하여 성능을 높이고 부하는 낮추는 장비인 웹 캐시가 동작하는 방식을 알아본다.
- 8장 통합점: 게이트웨이, 터널, 릴레이: 보안 소켓 계층(Secure Sockets Layer, SSL) 같이 HTTP가 아닌 프로토콜로 통신하는 소프트웨어가 HTTP를 사용해서 통신할 수 있게 해주는 게이트웨이와 애플리케이션 서버에 대해 설명한다.
- 9장 웹 로봇: 유비쿼터스 브라우저, 로봇, 스파이더, 검색엔진 같이 웹 전반에서 쓰이는 다양한 형태의 클라이언트에 대해 설명한다.
- 10장 HTTP/2.0: 현재 개발이 진행 중인 HTTP인 HTTP/2.0 프로토콜을 다룬다.<sup>1</sup>

3부 <식별, 인가, 보안>에서는 사용자 식별, 보안의 적용, 콘텐츠 접근 제어에 대한 기법 및 기술을 다룬다. 3부는 다음과 같은 내용을 포함한다.

- 11장 클라이언트 식별과 쿠키: 특정 사용자만 볼 수 있는 콘텐츠를 제공하는 데 쓰이는 사용자 식별 기술을 다룬다.
- 12장 기본 인증: 사용자를 식별하는 데 쓰이는 기본 체계를 주로 설명한다. 이 장에서는 데이터베이스에서 HTTP 인증이 어떻게 이루어지는지 알아볼 것이다.
- 13장 다이제스트 인증: 다이제스트 인증을 설명하고, HTTP의 보안을 강화하기 위해 제안된 개선사항들을 알아본다.
- 14장 보안 HTTP: 인터넷 암호화, 디지털 인증서, SSL에 대해 자세히 다룬다.

---

1 (옮긴이) 10장은 현재의 변화를 다루기 위해 옮긴이들이 새로 집필했다.



4부 <엔터티, 인코딩, 국제화>는, 메시지 본문에 있는 콘텐츠를 기술하고 생성하는 웹 표준과 HTTP 메시지 본문(실제 웹 콘텐츠를 담고 있는)을 주로 다룬다. 4부 내용은 다음과 같다.

- 15장 엔터티와 인코딩: HTTP 콘텐츠의 구조를 설명한다.
- 16장 국제화: 웹 콘텐츠를 전 세계 모든 사용자가 읽을 수 있도록, 다른 언어와 문자로 변환해주는 웹 표준에 대해 알아본다.
- 17장 내용 협상과 트랜스코딩: 적절한 콘텐츠를 받기 위한 협상 체계를 설명한다.

5부 <콘텐츠 발행 및 배포>는, 웹 콘텐츠를 발행하고 배포하는 기술에 대해 논의한다. 5부는 다음과 같은 내용을 포함한다.

- 18장 웹 호스팅: 현대 웹 호스팅 환경에 있는 서버에 배포하는 방법과 HTTP가 가상 웹 호스팅을 지원하는 방식에 대해 다룬다.
- 19장 배포 시스템: 웹 콘텐츠를 생성하고 웹 서버에 그 콘텐츠를 배포하는 기술에 대해 논의한다.
- 20장 리다이렉션과 부하 균형: 유입되는 웹 트래픽을 서버군에 분배하는 기술과 도구 들에 대해 다룬다.
- 21장 로깅과 사용 추적: 로그 포맷과 그에 대한 일반적인 질문들을 다룬다.

6부 <부록>은 도움이 될 만한 참고자료와 기술 관련 튜토리얼을 제공한다.

- 부록 A. URI 스킴: 통합 자원 식별자(Uniform Resource Identifier, URI) 스킴을 통해 지원하는 프로토콜 요약했다.
- 부록 B. HTTP 상태 코드: HTTP 응답 코드 목록을 편히 볼 수 있게 제공한다.
- 부록 C. HTTP 헤더: HTTP 헤더 필드의 목록을 제공한다.
- 부록 D. MIME 타입: 광범위한 MIME 타입 목록을 제공하며 MIME 타입을 어떻게 등록하는지 설명한다.
- 부록 E. base-64 인코딩: HTTP 인증에서 사용하는 base-64 인코딩을 설명한다.
- 부록 F. 다이제스트 인증: HTTP에서 다양한 인증 스킴을 구현하는 방법에 대해 자세히 설명한다.
- 부록 G. 언어 태그: HTTP의 언어 관련 헤더들에서 사용하는 언어 태그값을 설

명한다.

- 부록 H. MIME 캐릭터셋 등록: HTTP 국제화 지원에 사용하는 문자 인코딩에 대한 상세한 목록을 제공한다.

각 장은 풍부한 예와 추가로 참고할 만한 자료를 담고 있다.

## 의견과 질문

이 책에 대한 의견이나 질문이 있으면 출판사에 연락하기 바란다.

O'Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (in the United States or Canada) (707) 829-0515 (international/  
local)

(707) 829-0104 (fax)

다음은 이 책에 대한 오탈자, 예, 추가 정보가 있는 웹페이지다.<sup>2</sup>

<http://www.oreilly.com/catalog/htptdg/>

이 책에 대한 기술적인 질문이나 의견은 다음 이메일로 보내주기 바란다.

[bookquestions@oreilly.com](mailto:bookquestions@oreilly.com)

책, 콘퍼런스, 리스소 센터, 오라일리 네트워크에 대한 정보는 다음 오라일리 웹 사이트를 방문하기 바란다.

<http://www.oreilly.com>

---

2 (굵긴이) 번역서의 오탈자는 <http://www.insightbook.co.kr/65221>에 등록되어 있다.

## 감사의 말

이 책은 많은 사람의 노력으로 만들어졌다. 5명의 저자는 이 책에 큰 도움을 준 사람들에게 감사의 마음을 전하고자 한다.

우선 오라일리의 편집자인 린다 무이(Linda Mui)에게 감사의 뜻을 표한다. 린다는 1996년에 데이비드(David)와 브라이언(Brian)을 처음 만났으며, 이 책을 좀 더 세련되고 정교하게 다듬어 주었다. 또한 린다는 책 저술은 처음이었던 우리에게, 이 책이 일관된 방향성을 갖도록 이끌었으며 일정을 맞추는 데 필요한 시간관리도 해주었다. 무엇보다도, 린다는 이 책을 집필할 기회를 주었다. 다시 한번 그녀에게 큰 고마움을 전한다.

또한, 이 책을 검토하고 의견을 주고 초안을 수정해 주는데 많은 힘을 써준, 명석하고 박식하며 친절한 영혼들인 토니 보크(Tony Bourke), 신 버크(Sean Burke), 마이크 코우라(Mike Chowla), 샤이나즈 데이버(Shernaz Daver), 프레드 더글리스(Fred Douglass), 파울라 퍼거슨(Paula Ferguson), 비카스 자(Vikas Jha), 예브스 라폰(Yves Lafon), 피터 매티스(Peter Mattis), 척 니어다일(Chuck Neerdaels), 루이스 태버라(Luis Tavera), 다인 웨슬(Duane Wessels), 데이브 우(Dave Wu), 마르코 자가(Marco Zaghera)에게 감사의 뜻을 표한다. 그들의 견해와 제안은 이 책을 개선하는데 큰 도움이 되었다.

이 책에 있는 멋진 그림들 대부분은 오라일리의 랍 로만도(Rob Romano)가 그렸다. 이 책은 어려운 개념을 쉽게 이해하는데 도움이 되는 정교한 그림이 매우 많다. 이 많은 그림은 공들여 그려졌으며 여러 번의 교정을 거쳤다. 만약 그림이 수천 단어보다 더 가치가 있다 한다면, 랍은 이 책에 수백 쪽에 달하는 가치를 더해 주었다고 생각한다.

브라이언은 이 책에 공헌한 모든 저자에게 감사의 뜻을 표한다. HTTP를 자세하게 다루는 동시에 이해하기 쉽도록 책을 집필하기 위해 모든 저자가 많은 시간을 투자하였다. 결혼식, 아이들의 생일, 살인적인 업무, 스타트업, 학교 졸업 등 많은 일이 있었음에도, 저자들은 이 프로젝트를 성공적으로 마쳤다. 이 책이 그 모든 사람들이 고생한 결과라고 생각하며, 무엇보다도 덕분에 아주 좋은 책이 나올 수 있게 되었다. 브라이언은 인크토미(Inktomi) 직장 동료들의 열의와 지원, 그리고 실제 애플리케이션에서 HTTP를 사용하는 것에 대한 깊은 통찰에 감사의 마음을 전한다. 또한, 이 책에서 설명을 돕기 위해 Cajun-shop.com을 사용하도록 허락해준 Cajun-shop 친구들에게도 감사의 마음을 전한다.

데이비드는 그의 가족, 특히 어머니와 할아버지의 지속적인 지원에 감사함을 전한다. 그들은 데이비드가 책을 쓰는 몇 년간의 불규칙했던 생활을 이해해줬다. 또한 슬럽(Slurp), 오르토미(Orctomi), 노마(Norma)와 동료 저자들의 노고에도 감사의 마음을 전한다. 마지막으로, 이런 새로운 경험을 하게 해준 브라이언에게 감사의 뜻을 표한다.

마조리(Marjorie)는 그동안 많은 이해와 지원 및 기술적인 통찰을 제시해주었던, 그녀의 남편인 알란 리우(Alan Liu)에게 감사의 뜻을 표한다. 마조리는 수많은 통찰과 영감을 주었던 동료 저자들에게도 감사의 뜻을 표한다. 그녀는 그들과 함께 이 책을 만들었던 경험 역시 감사히 생각한다.

사이루(Sailu)는 이 책을 만들 기회를 준 데이비드와 브라이언에게, 그리고 HTTP를 소개해준 척 니어다일(Chuck NeerDael)에게 감사의 마음을 전한다.

안쉬우(Anshu)는 이 책을 쓰는 수년 동안 인내와 지원 및 격려를 아끼지 않았던 그의 아내 라시(Rashi)와 그의 부모님에게 감사의 마음을 전한다.

마지막으로, 지난 40년간 연구, 개발, 전파를 통해 우리의 과학, 사회, 경제 커뮤니티에 공헌해 왔던 이름난 그리고 이름 없는 인터넷 개척자들에게 우리 저자들 모두가 감사의 뜻을 표한다. 그들이 없었다면, 이 책도 없었을 것이다.

## HTTP: 웹의 기초

---

여기에서는 HTTP 프로토콜을 소개한다. 다음 4개의 장에서 웹의 기초가 되는 HTTP 핵심 기술을 설명한다.

- 1장 HTTP 개관에서는 HTTP를 빠르게 훑어본다.
- 2장 URL과 리소스에서는 URL의 포맷과 인터넷에 있는 URL들이 가리키는 다양한 리소스 형식에 대해 자세히 알아본다. 또한 URN으로 발전하는 과정에 대해서도 개략적으로 알아본다.
- 3장 HTTP 메시지에서는 웹 콘텐츠를 실어 나르는 HTTP 메시지에 대해 자세히 알아본다.
- 4장 커넥션 관리에서는 HTTP에서 관리하는 TCP 커넥션에 대한 일반적인 오해들과 잘못된 작성된 규칙 및 동작 방식에 대해서 알아본다.



---

# 1장

---

H T T P : T h e D e f i n i t i v e G u i d e

---

## HTTP 개관

---

전 세계의 웹브라우저, 서버, 웹 애플리케이션은 모두 HTTP(Hypertext Transfer Protocol)를 통해 서로 대화한다. HTTP는 현대 인터넷의 공용어이다.

이 장은 HTTP를 간결하게 설명한다. 독자들은 얼마나 많은 웹 애플리케이션이 HTTP를 이용해 통신하고, HTTP가 어떻게 그 일을 해내는지 개략적으로 알게 될 것이다. 특히 다음에 대해 이야기할 것이다.

- 얼마나 많은 클라이언트와 서버가 통신하는지
- 리소스(웹 콘텐츠)가 어디서 오는지
- 웹 트랜잭션이 어떻게 동작하는지
- HTTP 통신을 위해 사용하는 메시지의 형식
- HTTP 기저의 TCP 네트워크 전송
- 여러 종류의 HTTP 프로토콜
- 인터넷 곳곳에 설치된 다양한 HTTP 구성요소

공부할 거리를 충분히 확보했으니, 이제 HTTP의 세계로 여행을 떠나보자.

### 1.1 HTTP: 인터넷의 멀티미디어 배달부

수십억 개의 JPEG 이미지, HTML 페이지, 텍스트 파일, MPEG 동영상, WAV 음성 파일, 자바 애플릿 등이 하루도 쉬지 않고 인터넷을 향해한다. HTTP는 전 세계의 웹 서버로부터 이 대량의 정보를 빠르고, 간편하고, 정확하게 사람들의 PC에 설치

된 웹브라우저로 옮겨준다.

HTTP는 신뢰성 있는 데이터 전송 프로토콜을 사용하기 때문에, 데이터가 지구 반대편에서 오더라도 전송 중 손상되거나 꼬이지 않음을 보장한다. 이 덕분에 사용자는 인터넷에서 얻은 정보가 손상된 게 아닌지 염려하지 않아도 된다. 신뢰성 있는 전송은 인터넷 애플리케이션 개발자에게도 이로운데, HTTP 통신이 전송 중 파괴되거나, 중복되거나, 왜곡되는 것을 걱정하지 않아도 되기 때문이다. 개발자는 인터넷의 결함이나 약점에 대한 걱정 없이 애플리케이션 고유의 기능을 구현하는데 집중할 수 있다.

HTTP가 웹 트래픽을 어떻게 전송하는지 더 자세히 알아보자.

## 1.2 웹 클라이언트와 서버

웹 콘텐츠는 웹 서버에 존재한다. 웹 서버는 HTTP 프로토콜로 의사소통하기 때문에 보통 HTTP 서버라고 불린다. 이들 웹 서버는 인터넷의 데이터를 저장하고, HTTP 클라이언트가 요청한 데이터를 제공한다. 그림 1-1에 그려진 대로, 클라이언트는 서버에게 HTTP 요청을 보내고 서버는 요청된 데이터를 HTTP 응답으로 돌려준다. HTTP 클라이언트와 HTTP 서버는 월드 와이드 웹의 기본 요소다.

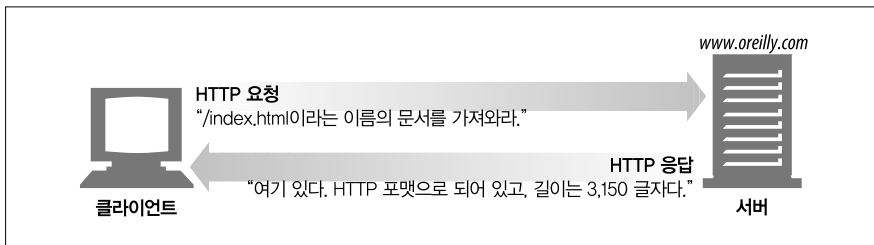


그림 1-1 웹 클라이언트와 웹 서버

아마 독자들은 HTTP 클라이언트를 매일 이용하고 있을 것이다. 가장 흔한 클라이언트는 마이크로소프트 인터넷 익스플로러나 구글 크롬 같은 웹브라우저다. 웹브라우저는 서버에게 HTTP 객체를 요청하고 사용자의 화면에 보여준다.

예를 들어 “http://www.oreilly.com/index.html” 페이지를 열어볼 때, 웹브라우저는 HTTP 요청을 www.oreilly.com 서버로 보낸다(그림 1-1 참고). 서버는 요청 받은 객체(이 경우 “/index.html”)를 찾고, 성공했다면 그것의 타입, 길이 등의 정보



와 함께 HTTP 응답에 실어서 클라이언트에게 보낸다.

### 1.3 리소스

웹 서버는 웹 리소스를 관리하고 제공한다. 웹 리소스는 웹 콘텐츠의 원천이다. 가장 단순한 웹 리소스는 웹 서버 파일 시스템의 정적 파일이다. 정적 파일은 텍스트 파일, HTML 파일, 마이크로소프트 워드 파일, 어도비 아크로벳 파일, JPEG 이미지 파일, AVI 동영상 파일, 그 외 모든 종류의 파일을 포함한다.

그러나 리소스는 반드시 정적 파일이어야 할 필요는 없다. 리소스는 요청에 따라 콘텐츠를 생산하는 프로그램이 될 수도 있다. 이들 동적 콘텐츠 리소스는 사용자가 누구인지, 어떤 정보를 요청했는지, 몇 시인지에 따라 다른 콘텐츠를 생성한다. 또 카메라에서 라이브 영상을 가져와 보여주거나, 주식 거래, 부동산 데이터베이스 검색, 온라인 쇼핑몰에서 선물 구입을 할 수 있게 해줄 수도 있다(그림 1-2 참조).

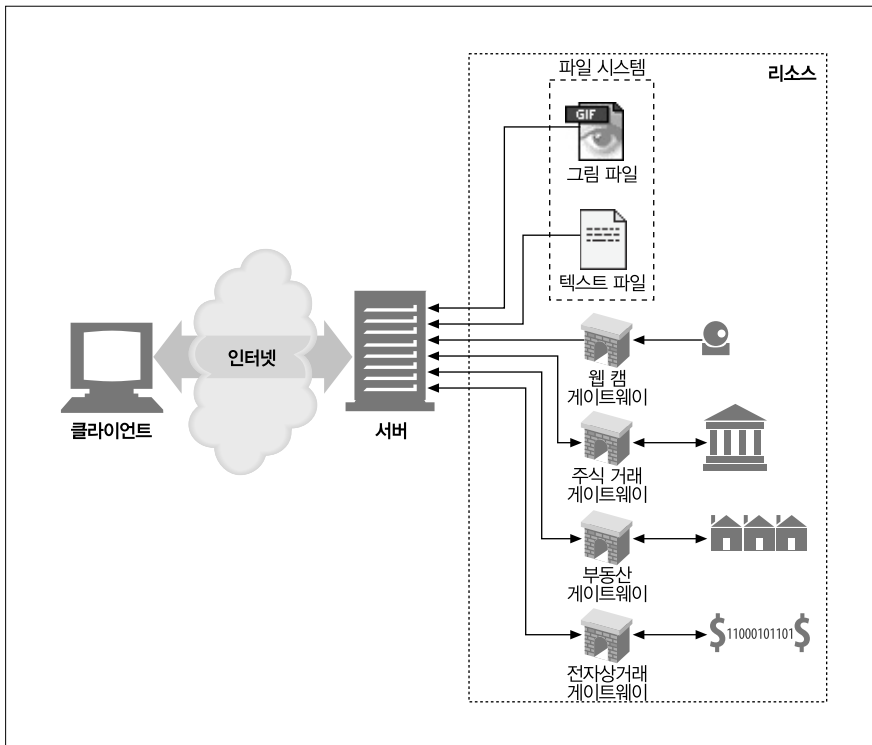


그림 1-2 웹 리소스란 웹에 콘텐츠를 제공하는 모든 것을 말한다.

요약하자면, 어떤 종류의 콘텐츠 소스도 리소스가 될 수 있다. 예컨대 기업 판매에

즉 스프레드시트 파일은 리소스다. 지역 공공 도서관의 서가를 탐색하는 웹 게이트 웨이도 리소스다. 인터넷 검색엔진 역시 리소스다.

### 1.3.1 미디어 타입

인터넷은 수천 가지 데이터 타입을 다루기 때문에, HTTP는 웹에서 전송되는 객체 각각에 신중하게 MIME 타입이라는 데이터 포맷 라벨을 붙인다. MIME (Multipurpose Internet Mail Extensions, 다목적 인터넷 메일 확장)은 원래 각기 다른 전자메일 시스템 사이에서 메시지가 오갈 때 겪는 문제점을 해결하기 위해 설계되었다. MIME은 이메일에서 워낙 잘 동작했기 때문에, HTTP에서도 멀티미디어 콘텐츠를 기술하고 라벨을 붙이기 위해 채택되었다.

웹 서버는 모든 HTTP 객체 데이터에 MIME 타입을 붙인다(그림 1-3). 웹브라우저는 서버로부터 객체를 돌려받을 때, 다룰 수 있는 객체인지 MIME 타입을 통해 확인한다. 대부분의 웹브라우저는 잘 알려진 객체 타입 수백 가지를 다룰 수 있다. 이미지 파일을 보여주고, HTML 파일을 분석하거나 포맷팅하고, 오디오 파일을 컴퓨터의 스피커를 통해 재생하고, 특별한 포맷의 파일을 다루기 위해 외부 플러그인 소프트웨어를 실행한다.

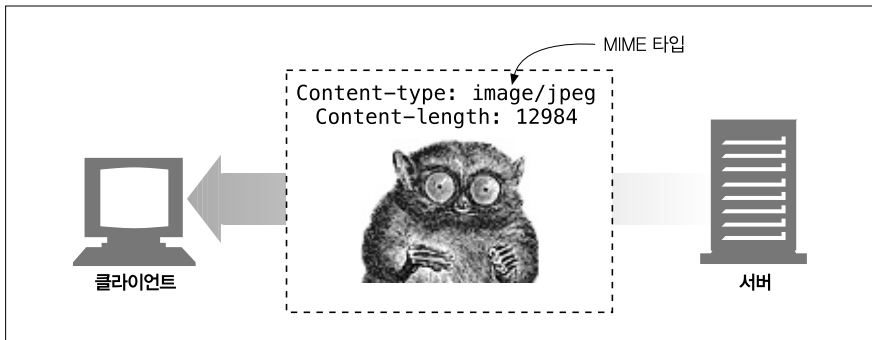


그림 1-3 웹 서버는 데이터 콘텐츠와 함께 MIME 타입을 보내준다.

MIME 타입은 사선(/)으로 구분된 주 타입(primary object type)과 부 타입(specific subtype)으로 이루어진 문자열 라벨이다. 예를 들면 다음과 같다.

- HTML로 작성된 텍스트 문서는 text/html 라벨이 붙는다.
- plain ASCII 텍스트 문서는 text/plain 라벨이 붙는다.
- JPEG 이미지는 image/jpeg가 붙는다.
- GIF 이미지는 image/gif가 된다.

- 애플 킷타임 동영상은 video/quicktime이 붙는다.
- 마이크로소프트 파워포인트 프레젠테이션은 application/vnd.ms-powerpoint가 붙는다.

수백 가지의 잘 알려진 MIME 타입과, 그보다 더 많은 실험용 혹은 특정 용도의 MIME 타입이 존재한다. MIME 타입 전체 목록은 부록 D에 실려 있다.

### 1.3.2 URI

웹 서버 리소스는 각자 이름을 갖고 있기 때문에, 클라이언트는 관심 있는 리소스를 지목할 수 있다. 서버 리소스 이름은 통합 자원 식별자(uniform resource identifier), 혹은 URI로 불린다. URI는 인터넷의 우편물 주소 같은 것으로, 정보 리소스를 고유하게 식별하고 위치를 지정할 수 있다.

‘쥘의 컴퓨터 가게’의 웹 서버에 있는 이미지 리소스에 대한 URI라면 이런 식이다.

`http://www.joes-hardware.com/specials/saw-blade.gif`

그림 1-4는 쥘의 컴퓨터 가게 서버에 있는 GIF 형식의 톱날 그림 리소스에 대한 URI가 HTTP 프로토콜에서 어떻게 해석되는지 보여준다. HTTP는 주어진 URI로 객체를 찾아온다. URI에는 두 가지가 있는데, URL과 URN이라는 것이다. 이 두 종류의 자원 식별자에 대해 지금 살펴보자.

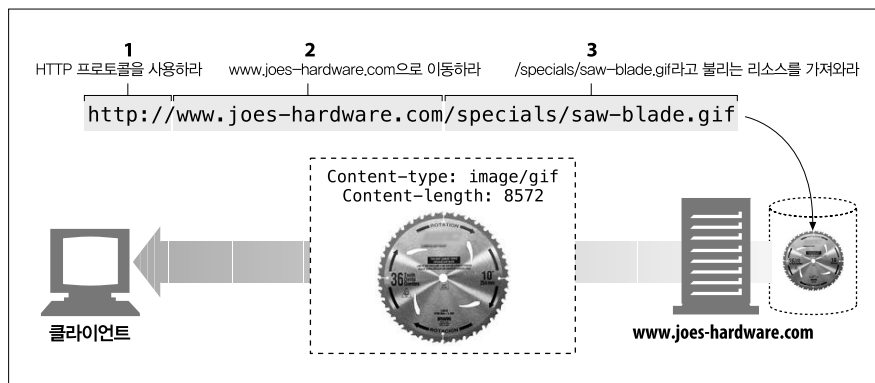


그림 1-4 URL은 프로토콜, 서버, 리소스를 명시한다.

### 1.3.3 URL

통합 자원 지시자(uniform resource locator, URL)는 리소스 식별자의 가장 흔한 형태다. URL은 특정 서버의 한 리소스에 대한 구체적인 위치를 서술한다. URL은 리소스가 정확히 어디에 있고 어떻게 접근할 수 있는지 분명히 알려준다. 그림 1-4는 어떻게 URL로 리소스의 정확한 위치와 접근방법을 표현하는지 보여준다. 표 1-1은 URL의 몇 가지 예다.

URL	설명
<a href="http://www.oreilly.com/index.html">http://www.oreilly.com/index.html</a>	오라일리 출판사 홈페이지의 URL
<a href="http://www.yahoo.com/images/logo.gif">http://www.yahoo.com/images/logo.gif</a>	야후! 웹 사이트 로고의 URL
<a href="http://www.joes-hardware.com/inventory-check.cgi?item=12731">http://www.joes-hardware.com/inventory-check.cgi?item=12731</a>	물품 #12731의 재고가 있는지 확인하는 프로그램에 대한 URL
<a href="ftp://joe:tools4u@ftp.joes-hardware.com/locking-pliers.gif">ftp://joe:tools4u@ftp.joes-hardware.com/locking-pliers.gif</a>	비밀번호로 보호되는 FTP를 통해 locking-pliers.gif 이 미지 파일에 접근하는 URL

표 1-1 URL의 예

대부분의 URL은 세 부분으로 이루어진 표준 포맷을 따른다.

- URL의 첫 번째 부분은 스킴(scheme)이라고 불리는데, 리소스에 접근하기 위해 사용되는 프로토콜을 서술한다. 보통 HTTP 프로토콜(<http://>)이다.
- 두 번째 부분은 서버의 인터넷 주소를 제공한다(예: [www.joes-hardware.com](http://www.joes-hardware.com)).
- 마지막은 웹 서버의 리소스를 가리킨다(예: [/specials/saw-blade.gif](http://www.joes-hardware.com/specials/saw-blade.gif)).

오늘날 대부분의 URI는 URL이다.

### 1.3.4 URN

URI의 두 번째 종류는 유니폼 리소스 이름(uniform resource name, URN)이다. URN은 콘텐츠를 이루는 한 리소스에 대해, 그 리소스의 위치에 영향 받지 않는 유일무이한 이름 역할을 한다. 이 위치 독립적인 URN은 리소스를 여기저기로 옮기더라도 문제없이 동작한다. 리소스가 그 이름을 변하지 않게 유지하는 한, 여러 종류의 네트워크 접속 프로토콜로 접근해도 문제없다.

예를 들어, 다음의 URN은 인터넷 표준 문서 ‘RFC 2141’가 어디에 있거나 상관없이(심지어 여러 군데에 복사되었더라도) 그것을 지칭하기 위해 사용할 수 있다.

```
urn:ietf:rfc:2141
```

URN은 여전히 실험 중인 상태고 아직 널리 채택되지 않았다. 효율적인 동작을 위해 URN은 리소스 위치를 분석하기 위한 인프라 지원이 필요한데, 그러한 인프라가 부재하기에 URN 채택이 더 늦춰지고 있다. 그러나 URN의 전망은 분명 밝다. 우리는 2장에서 URN에 대해 조금 더 자세히 논의할 것이지만, 그 외 나머지 대부분에서는 거의 URL에만 초점을 맞출 것이다.

특별한 언급이 없으면, 앞으로는 통상적인 관례에 따라 URI와 URL을 같은 의미로 사용할 것이다.

## 1.4 트랜잭션

클라이언트가 웹 서버와 리소스를 주고받기 위해 HTTP를 어떻게 사용하는지 좀 더 자세히 알아보자. HTTP 트랜잭션은 요청 명령(클라이언트에서 서버로 보내는)과 응답 결과(서버가 클라이언트에게 돌려주는)로 구성되어 있다. 이 상호작용은 그림 1-5에 묘사된 것과 같이 HTTP 메시지라고 불리는 정형화된 데이터 덩어리를 이용해 이루어진다.

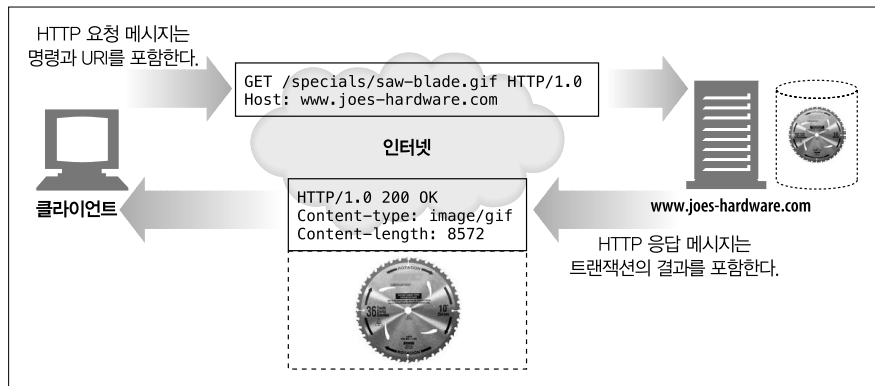


그림 1-5 HTTP 트랜잭션은 요청과 응답 메시지로 구성되어 있다.

### 1.4.1 메서드

HTTP는 HTTP 메서드라고 불리는 여러 가지 종류의 요청 명령을 지원한다. 모든 HTTP 요청 메시지는 한 개의 메서드를 갖는다. 메서드는 서버에게 어떤 동작이 취해져야 하는지 말해준다(웹페이지 가져오기, 게이트웨이 프로그램 실행하기, 파일 삭제하기 등). 표 1-2는 흔히 쓰이는 HTTP 메서드 다섯 개를 열거하고 있다.

HTTP 메서드	설명
GET	서버에서 클라이언트로 지정한 리소스를 보내라.
PUT	클라이언트에서 서버로 보낸 데이터를 지정한 이름의 리소스로 저장하라.
DELETE	지정한 리소스를 서버에서 삭제하라.
POST	클라이언트 데이터를 서버 게이트웨이 애플리케이션으로 보내라.
HEAD	지정한 리소스에 대한 응답에서, HTTP 헤더 부분만 보내라.

표 1-2 흔히 쓰이는 HTTP 메서드들

HTTP 메서드에 대해서는 3장에서 상세히 다룰 것이다.

### 1.4.2 상태 코드

모든 HTTP 응답 메시지는 상태 코드와 함께 반환된다. 상태 코드는 클라이언트에게 요청이 성공했는지 아니면 추가 조치가 필요한지 알려주는 세 자리 숫자다. 흔히 쓰이는 상태 코드 몇 가지가 표 1-3에 나와 있다.

HTTP 상태 코드	설명
200	좋다. 문서가 바르게 반환되었다.
302	다시 보내라. 다른 곳에 가서 리소스를 가져가라.
404	없음. 리소스를 찾을 수 없다.

표 1-3 흔히 쓰이는 HTTP 상태 코드 몇 가지

HTTP는 각 숫자 상태 코드에 텍스트로 된 “사유 구절(reason phrase)”도 함께 보낸다(그림 1-5의 응답 메시지 참조). 이 구문은 단지 설명만을 위해서 포함된 것일 뿐 실제 응답 처리에는 숫자로 된 코드가 사용된다.

HTTP 소프트웨어는 다음에 열거된 상태 코드와 사유 구절을 모두 같은 것으로 취급한다.

```
200 OK
200 Document attached
200 Success
200 All's cool, dude
```

HTTP 상태 코드는 3장에서 자세히 설명한다.

### 1.4.3 웹페이지는 여러 객체로 이루어질 수 있다

애플리케이션은 보통 하나의 작업을 수행하기 위해 여러 HTTP 트랜잭션을 수행