M02-03 · GIBALJA VISUAL MANUAL

HTTP 요청·응답과 상태 코드 읽기

> **한 문장 목표:** Network에서 한 요청과 응답을 골라 “무엇을 요청했고, 결과가 어땠으며, 다음에 무엇을 확인할지” 설명합니다.

난이도	개념	실습	셀프 테스트	최종 산출물
Level 1	55분	45분	15분	요청·응답 해부표, 상태 코드 판단표, 네트워크 분석 기록

상태 코드 백 개를 외우지 않습니다. 요청의 메서드·주소·헤더·본문과 응답의 상태·헤더·본문을 같은 한 쌍으로 읽고, 숫자를 다음 확인 행동으로 바꾸는 훈련을 합니다.

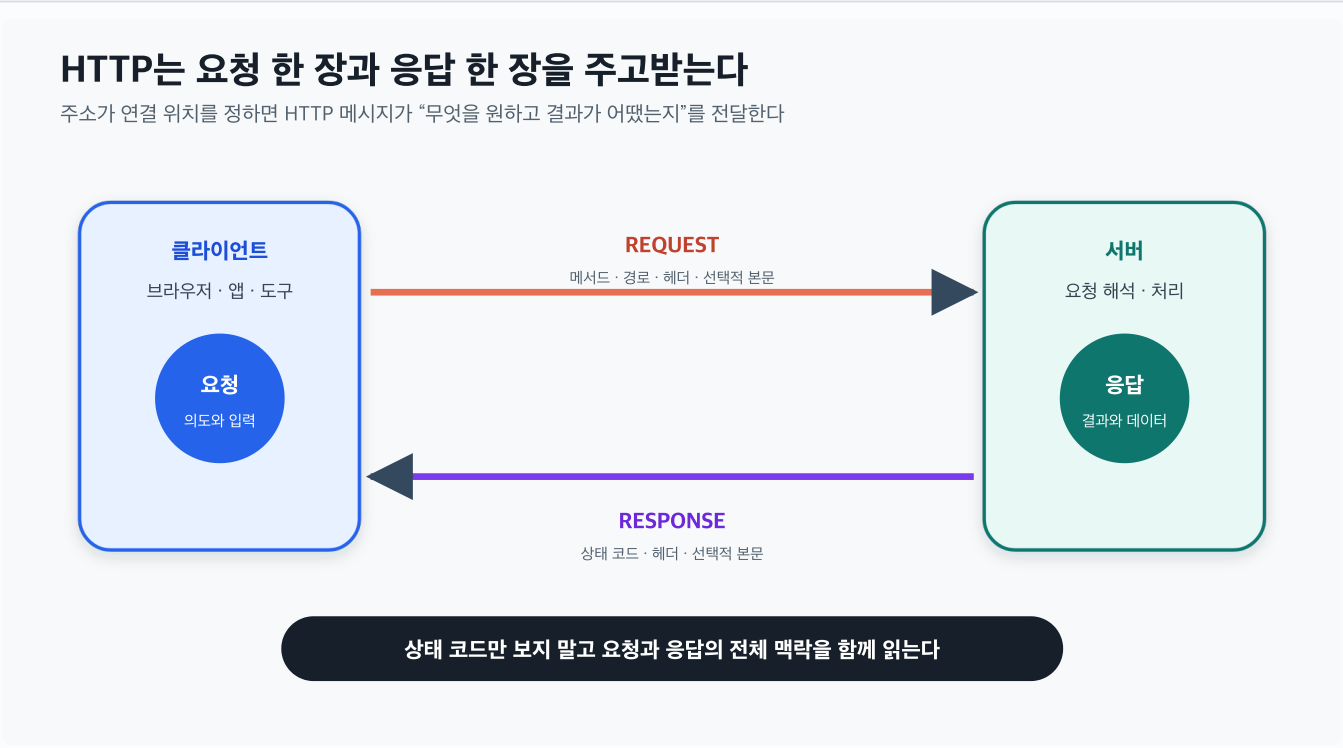
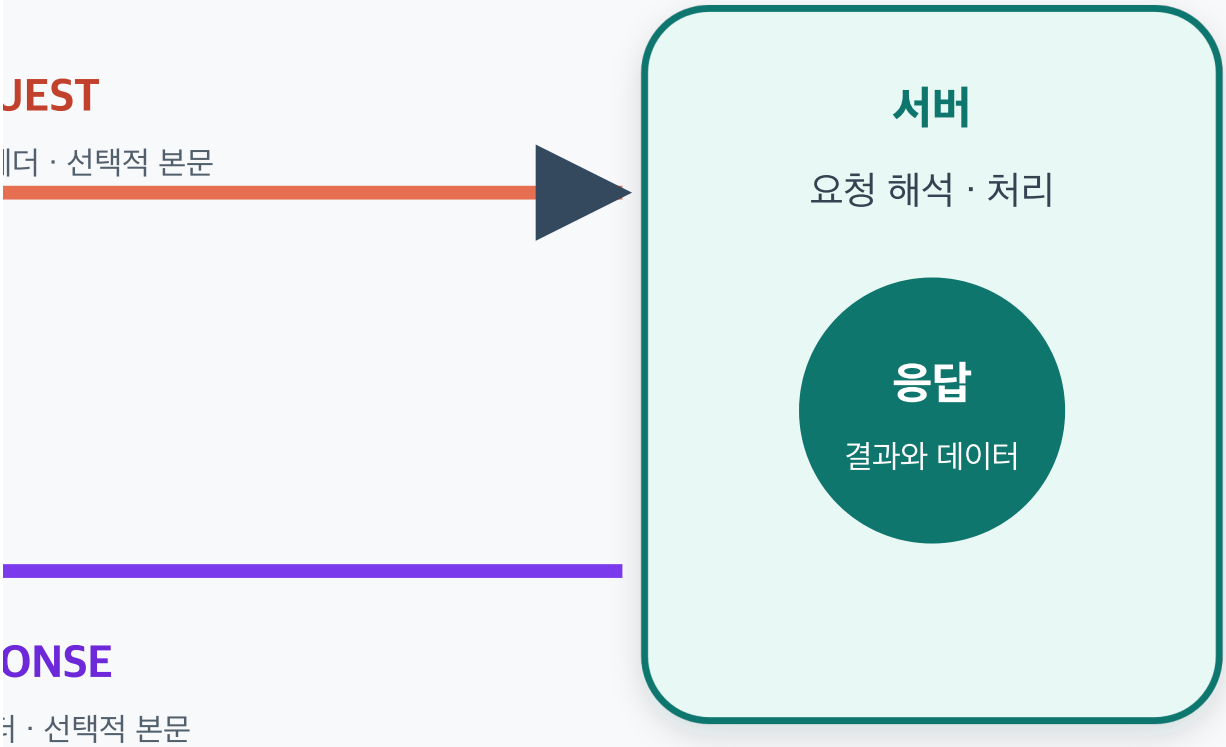


그림 1. HTTP는 클라이언트가 보낸 요청과 서버가 돌려준 응답을 한 쌍으로 읽어야 합니다.


클라이언트
브라우저 · 앱 · 도구
요청
의도와 입력
REQU
메서드 · 경로 · 허
RESP
상태 코드 · 헤더
상태 코드만 보지 말고 요청과 응



응답의 전체 맥락을 함께 읽는다

1. 이 PDF를 공부하는 방법

1회차: 요청과 응답의 네 부분 찾기 · 20분

그림 2와 그림 3을 보며 시작줄·헤더·빈 줄·본문을 다른 색으로 표시합니다.

2회차: 상태 코드를 행동으로 바꾸기 · 35분

상태 코드의 첫 숫자로 계열을 고르고, 자주 쓰는 코드에서 다음 확인 항목을 말합니다.

3회차: Network에서 실제 증거 수집하기 · 45분

`GET`, 폼 `POST`, 리다이렉션, `404`, `503`을 재현하고 요청·응답 해부표를 완성합니다.

4회차: 셀프 테스트 · 15분

정답을 가린 채 처음 보는 요청과 응답을 읽고, 개발자에게 전달할 문장을 만듭니다.

학습 완료 기준
한 요청에서 URL·메서드·상태 코드·중요 헤더·본문을 찾고, “이 코드는 이런 계열이므로 다음에는 이것을 확인하겠다”고 설명할 수 있습니다.

2. HTTP는 요청과 응답을 주고받는 규칙입니다

하이퍼텍스트 전송 프로토콜(Hypertext Transfer Protocol, HTTP)은 클라이언트와 서버가 요청과 응답 메시지를 주고받는 응용 계층 프로토콜입니다.

- 클라이언트: 브라우저·모바일 앱·명령줄 도구처럼 요청을 시작하는 프로그램
- 서버: 요청을 받고 해석해 응답하는 프로그램
- 요청(Request): 원하는 행동과 대상·조건·입력
- 응답(Response): 처리 결과와 메타데이터·결과 데이터

M02-02에서 배운 URL·DNS·IP·포트가 “어디로 연결할지”를 정했다면, HTTP는 연결한 뒤 “무엇을 원하고 결과가 어땠는지”를 표현합니다.

BIG IDEA 01

요청과 응답은 서로 떨어진 기록이 아니라 하나의 업무 대화입니다.

HTTP/1.1 모양으로 배우는 이유

이 교재는 사람이 읽기 쉬운 HTTP/1.1 텍스트 모양을 해부 모형으로 사용합니다. HTTP/2와 HTTP/3는 실제 전송 표현이 다르지만, 메서드·헤더·상태 코드가 전달하는 핵심 의미는 이어집니다.

Chrome Network는 실제 전송 버전에 관계없이 요청 URL·메서드·상태·헤더·본문을 사람이 확인하기 쉬운 형태로 보여 줍니다.

3. 요청과 응답의 공통 뼈대

HTTP 메시지는 다음 네 구역으로 읽습니다.

순서	구역	요청에서	응답에서
1	시작줄	메서드·요청 대상·버전	버전·상태 코드·선택적 설명
2	헤더	요청 조건·본문 형식·인증 정보	결과 조건·본문 형식·캐시·추적 정보
3	빈 줄	헤더가 끝났다는 구분	헤더가 끝났다는 구분

순서	구역	요청에서	응답에서
4	선택적 본문	서버에 보낼 데이터	결과·오류 상세 데이터

본문은 항상 있는 것이 아닙니다. 요청의 의도와 응답 상태에 따라 본문이 없을 수 있으며, `204 No Content`처럼 본문이 없어야 의미가 맞는 응답도 있습니다.

30초 확인

시작줄과 헤더 사이, 헤더와 본문 사이를 구분하는 것은 무엇일까요?

정답 보기

시작줄 다음에는 헤더가 이어지고, 헤더가 끝난 뒤 빈 줄 하나가 본문의 시작을 구분합니다.

4. 요청을 해부합니다

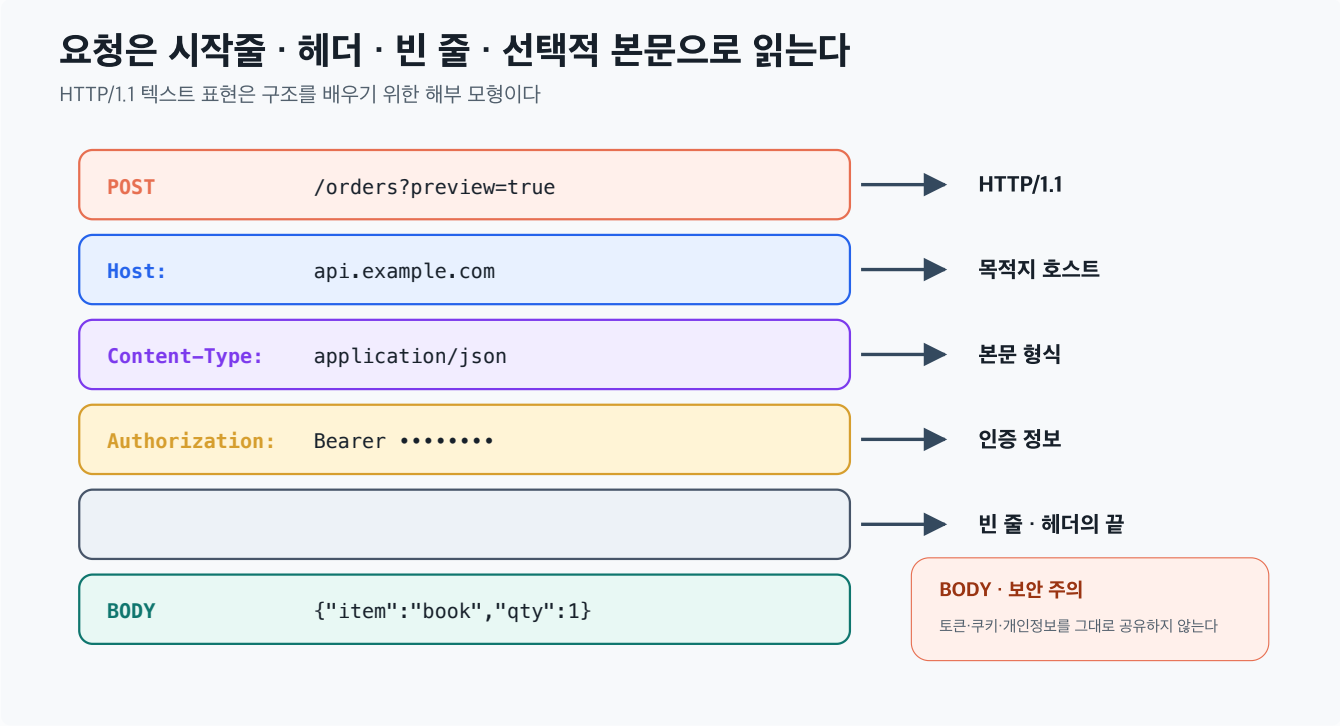
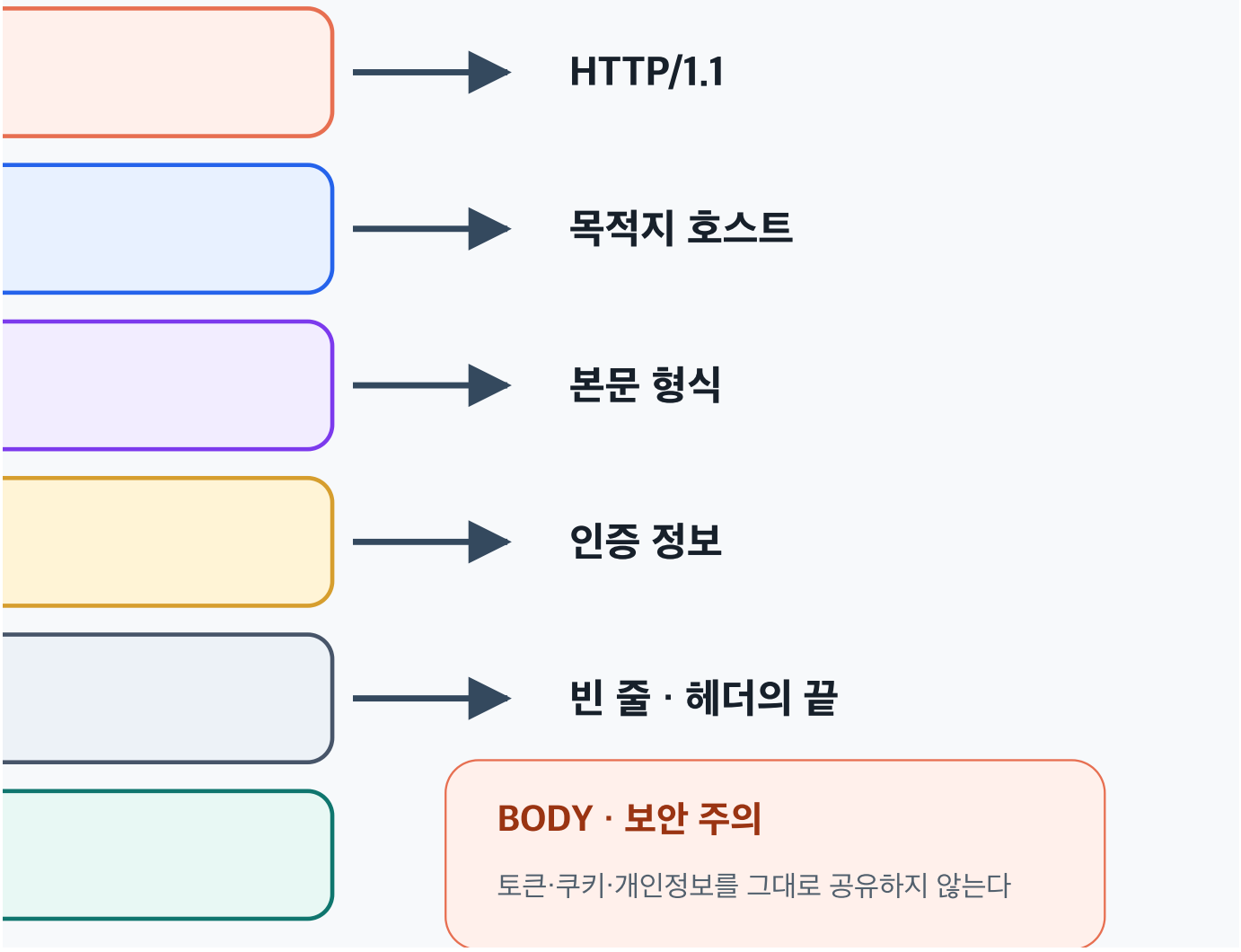


그림 2. 요청은 무엇을 할지, 어느 대상을 향하는지, 어떤 조건과 데이터를 보낼지 설명합니다.

```
POST          /orders?preview=true
Host:         api.example.com
Content-Type: application/json
Authorization: Bearer ••••••••

BODY          {"item":"book","qty":1}
```


HTTP/1.1
목적지 호스트
본문 형식
인증 정보
빈 줄 · 헤더의 끝
BODY · 보안 주의
토큰·쿠키·개인정보를 그대로 공유하지 않는다

예시 요청을 한 줄씩 읽어 봅시다.

```
POST /orders?preview=true HTTP/1.1
Host: api.example.com
Content-Type: application/json
Authorization: Bearer [REDACTED]

{"item":"book","qty":1}
```

4.1. 요청줄: 행동과 대상을 말합니다

- `POST` : 서버가 입력을 자신의 규칙에 따라 처리해 달라는 메서드
- `/orders?preview=true` : 요청 대상 경로와 질의
- `HTTP/1.1` : 이 텍스트 예시에서 사용하는 HTTP 버전

실제 HTTP/2·HTTP/3 메시지는 이 줄과 같은 텍스트 모양으로 전송되지 않지만, 메서드와 대상이라는 의미는 유지됩니다.

4.2. 요청 헤더: 처리 조건을 말합니다

- `Host` : 요청하는 호스트
- `Content-Type` : 보내는 본문의 형식
- `Authorization` : 서버가 인증에 사용할 자격 정보

헤더 이름은 대소문자를 구분하지 않지만 문서와 도구에서는 관례적인 표기를 따르는 편이 읽기 쉽습니다.

4.3. 요청 본문: 입력 데이터를 담습니다

예시 본문은 JSON 형식으로 상품과 수량을 전달합니다. `POST` , `PUT` , `PATCH` 에서 본문을 흔히 사용합니다.

`GET` 요청 본문의 의미는 일반적으로 정의되어 있지 않으므로, 서비스가 명시적으로 합의한 특별한 경우가 아니라면 질의나 별도의 메서드를 사용합니다.

> **화면 공유 주의:** `Authorization`, `Cookie`, 접근 토큰, 세션 값, 주민등록번호·이메일·전화번호 같은 개인정보는 캡처·문서·메신저에 그대로 남기지 않습니다.

5. 메서드는 원하는 행동을 나타냅니다

그림 3. 메서드는 행동의 의도를 표현하지만, 실제 허용 범위와 입력 규칙은 서비스 명세에서 확인합니다.

GET 조회

현재 표현을 받기

읽기 중심

HEAD 메티

본문 없이 헤더 확인

읽기 중심

PUT 전체 교체

대상의 상태를 생성·교체

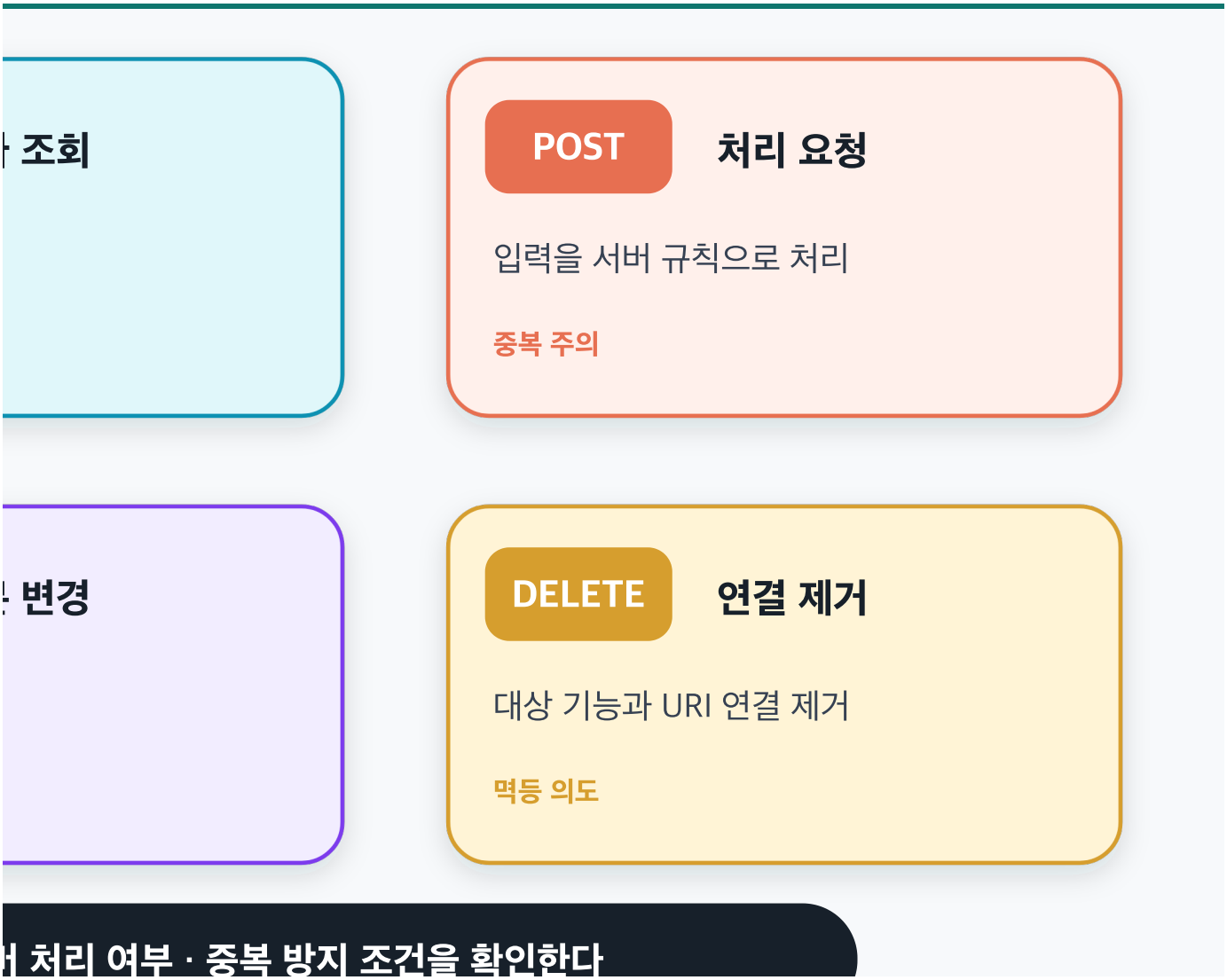
멱등 의도

PATCH 부분

일부 변경 지시

자동 반복 주의

실패 후 다시 보내기 전에는 메서드 · 서버


조회
POST
처리 요청
입력을 서버 규칙으로 처리
중복 주의
변경
DELETE
연결 제거
대상 기능과 URI 연결 제거
멱등 의도
처리 여부 · 중복 방지 조건을 확인한다

메서드	기본 의도	기발자 확인 질문
`GET`	현재 자원의 표현 조회	무엇을 조회하며 질의 조건은 무엇인가
`HEAD`	본문 없이 응답 헤더 조회	크기·수정 시각·존재 여부를 확인하려는가
`POST`	입력을 대상 자원의 규칙으로 처리	새 작업인가, 중복 요청에 안전한가
`PUT`	대상 자원의 상태를 생성하거나 전체 교체	전체 표현인가, 어느 자원을 정확히 지정했는가
`PATCH`	대상의 일부 변경	어떤 부분을 어떤 규칙으로 바꾸는가
`DELETE`	대상 URI와 현재 기능의 연결 제거 요청	삭제·비활성화·보관 중 실제 정책은 무엇인가

안전과 멱등은 같은 말이 아닙니다

- 안전한 메서드(Safe Method): 정의된 의도가 본질적으로 읽기 전용
- 멱등 메서드(Idempotent Method): 같은 요청을 여러 번 보낸 의도된 효과가 한 번 보낸 효과와 같음

`GET` · `HEAD` 는 안전한 메서드이고, 안전한 메서드와 `PUT` · `DELETE` 는 표준상 멱등 의도를 가집니다. 그러나 로그·알림 같은 부수 효과는 매번 생길 수 있고, 서비스가 잘못 구현되면 실제 결과도 달라질 수 있습니다.

`POST` 와 `PATCH` 를 통신 실패 뒤 무조건 다시 보내면 주문·결제·등록이 중복될 수 있습니다. 자동 재시도 전에 서버의 처리 여부와 중복 방지 키·업무 상태를 확인합니다.

30초 확인

결제 `POST` 를 보낸 직후 화면이 멈췄습니다. 같은 요청을 바로 다시 보내도 될까요?

정답 보기

바로 반복하지 않습니다. 첫 요청이 서버에서 처리됐는지, 주문·결제 상태와 요청 식별자, 중복 방지 장치가 있는지 먼저 확인합니다.

6. 헤더는 본문 바깥의 처리 메모입니다

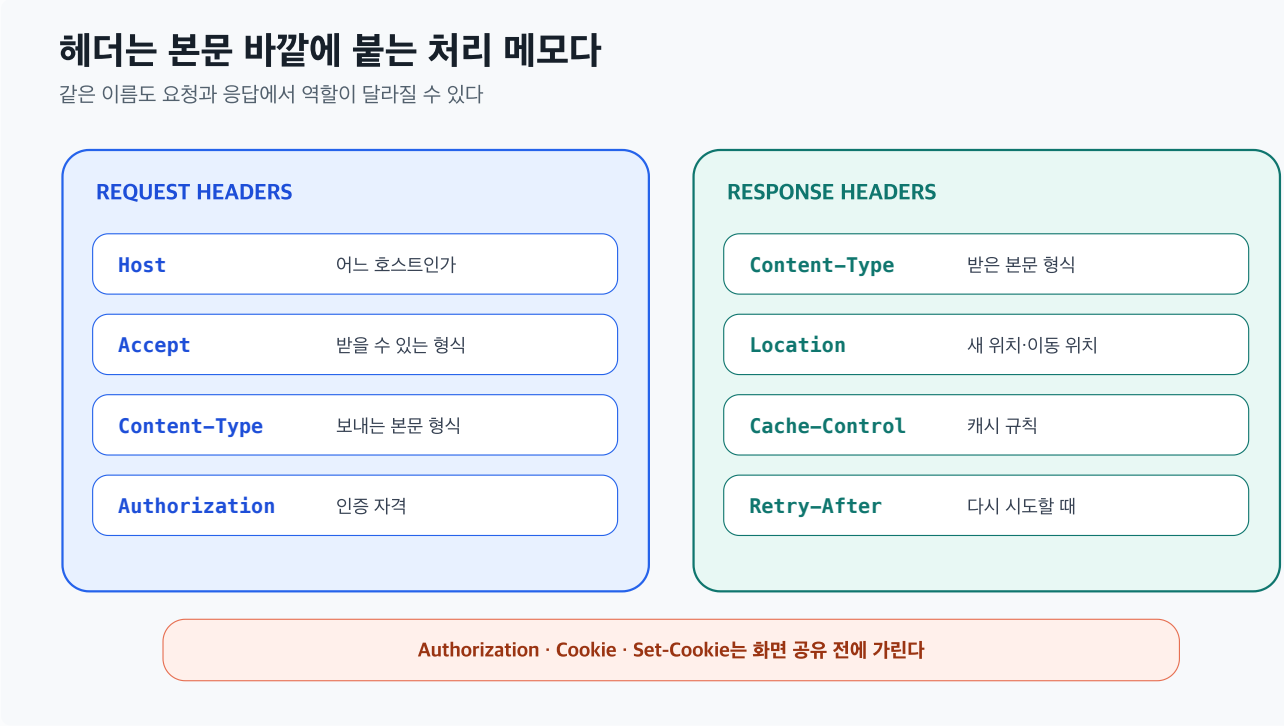


그림 4. 헤더는 본문 형식·인증·캐시·재시도처럼 메시지를 처리할 조건을 전달합니다.

REQUEST HEADERS

Host	어느 호스트인가
Accept	받을 수 있는 형식
Content-Type	보내는 본문 형식
Authorization	인증 자격

Authorization · Cookie · Set-

RESPONSE HEADERS

`Content-Type`	받은 본문 형식
`Location`	새 위치·이동 위치
`Cache-Control`	캐시 규칙
`Retry-After`	다시 시도할 때

Cookie는 화면 공유 전에 가린다

자주 보는 요청 헤더

헤더	역할	확인 질문
`Host`	대상 호스트	기대한 환경과 도메인인가
`Accept`	받을 수 있는 표현 형식	서버가 지원하는 형식과 맞는가
`Content-Type`	보내는 본문 형식	JSON·폼·파일 형식이 실제 본문과 맞는가
`Authorization`	인증 자격	값이 존재하고 만료되지 않았는가
`Cookie`	브라우저가 서버에 보내는 쿠키	세션·설정 값이 필요한 요청인가

자주 보는 응답 헤더

헤더	역할	확인 질문
`Content-Type`	받은 본문 형식	기대한 JSON·HTML·파일인가
`Location`	새 자원 또는 이동할 위치	`201` · `3xx` 뒤 어느 주소를 가리키는가
`Cache-Control`	캐시 가능 여부와 조건	최신 결과가 필요한 요청인가
`Retry-After`	다시 시도할 시각 또는 대기 시간	`429` · `503` 뒤 언제 재시도할 수 있는가
`Set-Cookie`	브라우저에 저장할 쿠키	보안 속성과 범위가 적절한가
`X-Request-Id` 등	요청 추적 식별자	서버 로그에서 같은 요청을 찾을 수 있는가

헤더 이름은 서비스마다 추가될 수 있습니다. 사용자 정의 추적 헤더의 실제 이름은 `X-Request-Id`, `Traceparent`, `Request-Id` 등으로 다를 수 있으므로 운영 기준을 확인합니다.

7. 응답을 해부합니다

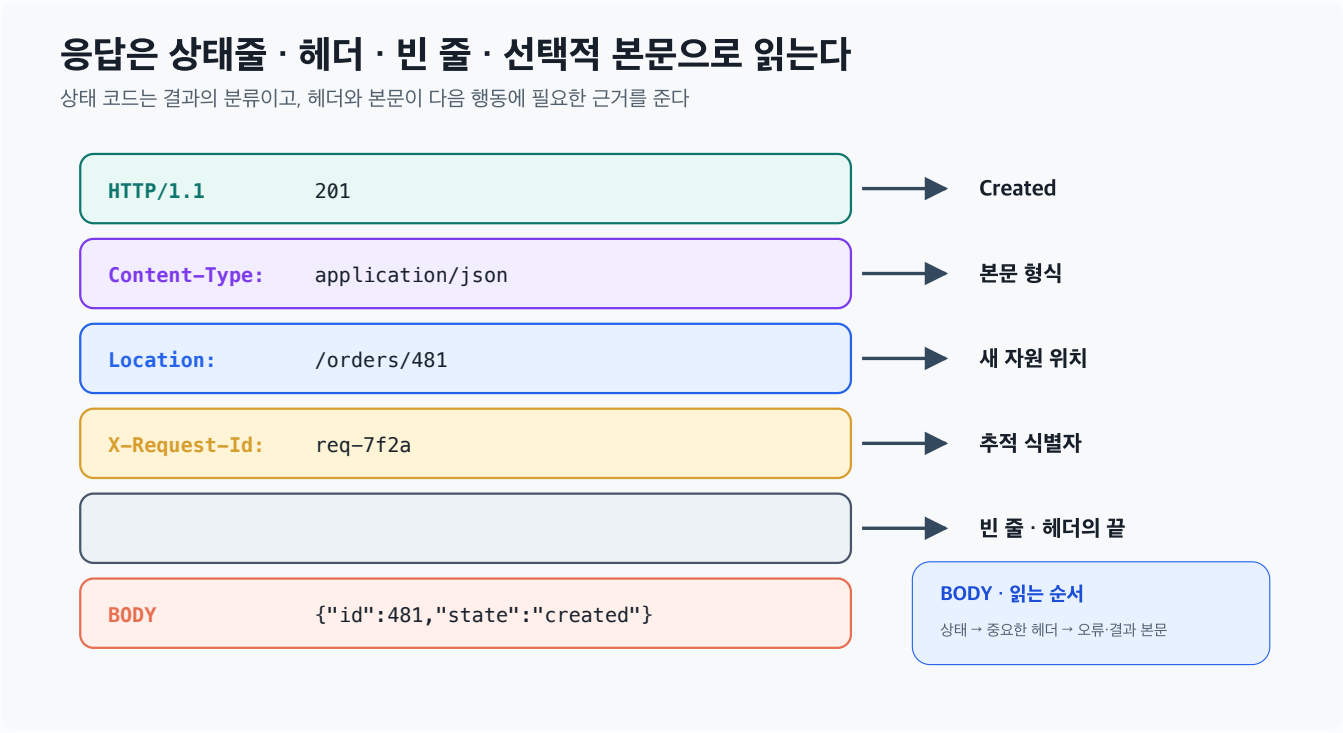
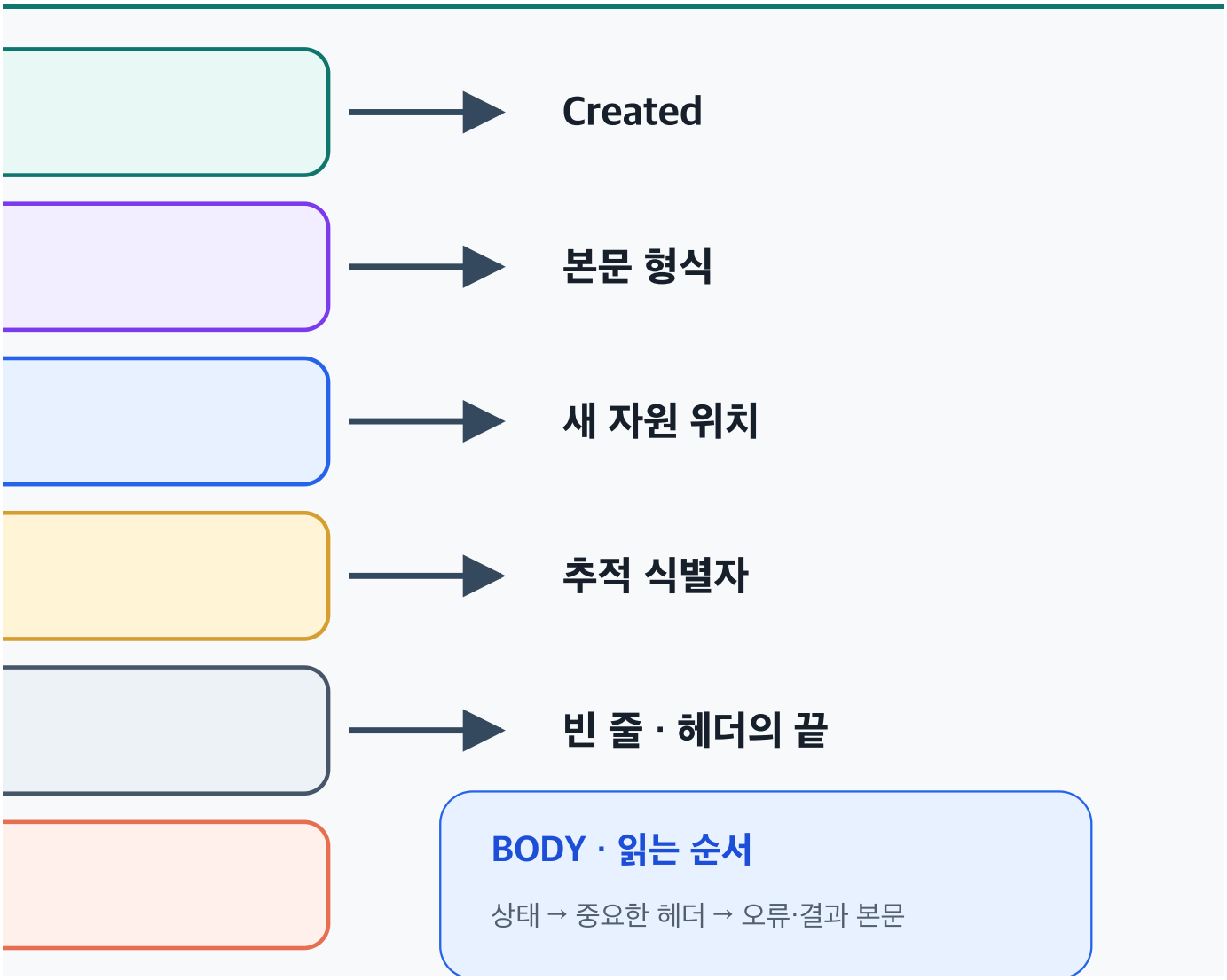


그림 5. 응답은 처리 결과를 상태 코드로 분류하고, 헤더와 본문으로 상세 정보를 제공합니다.

```
HTTP/1.1          201
Content-Type:     application/json
Location:         /orders/481
X-Request-Id:     req-7f2a

BODY              {"id":481,"state":"created"}
```


Created
본문 형식
새 자원 위치
추적 식별자
빈 줄 · 헤더의 끝
BODY · 읽는 순서
상태 → 중요한 헤더 → 오류·결과 본문

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: /orders/481
X-Request-Id: req-7f2a

{"id":481,"state":"created"}
```

7.1. 상태줄

- `HTTP/1.1` : 예시의 프로토콜 버전
- `201` : 요청 결과를 나타내는 세 자리 상태 코드
- `Created` : 사람이 읽도록 돕는 선택적 설명

프로그램의 판단은 숫자 상태 코드를 기준으로 합니다. 설명 문구는 생략되거나 다르게 표현될 수 있습니다.

7.2. 응답 헤더

`Content-Type`은 본문 형식을, `Location`은 새로 만들어진 자원의 위치를, 추적 식별자는 서버 로그에서 같은 요청을 찾을 단서를 제공합니다.

7.3. 응답 본문

성공 결과·오류 설명·필드별 검증 결과 등이 들어갈 수 있습니다. 다만 모든 응답에 본문이 있는 것은 아닙니다.

BIG IDEA 02

상태 코드는 결론의 분류이고, 헤더와 본문은 이유와 다음 행동의 근거입니다.

8. 상태 코드의 첫 숫자로 큰 방향을 잡습니다

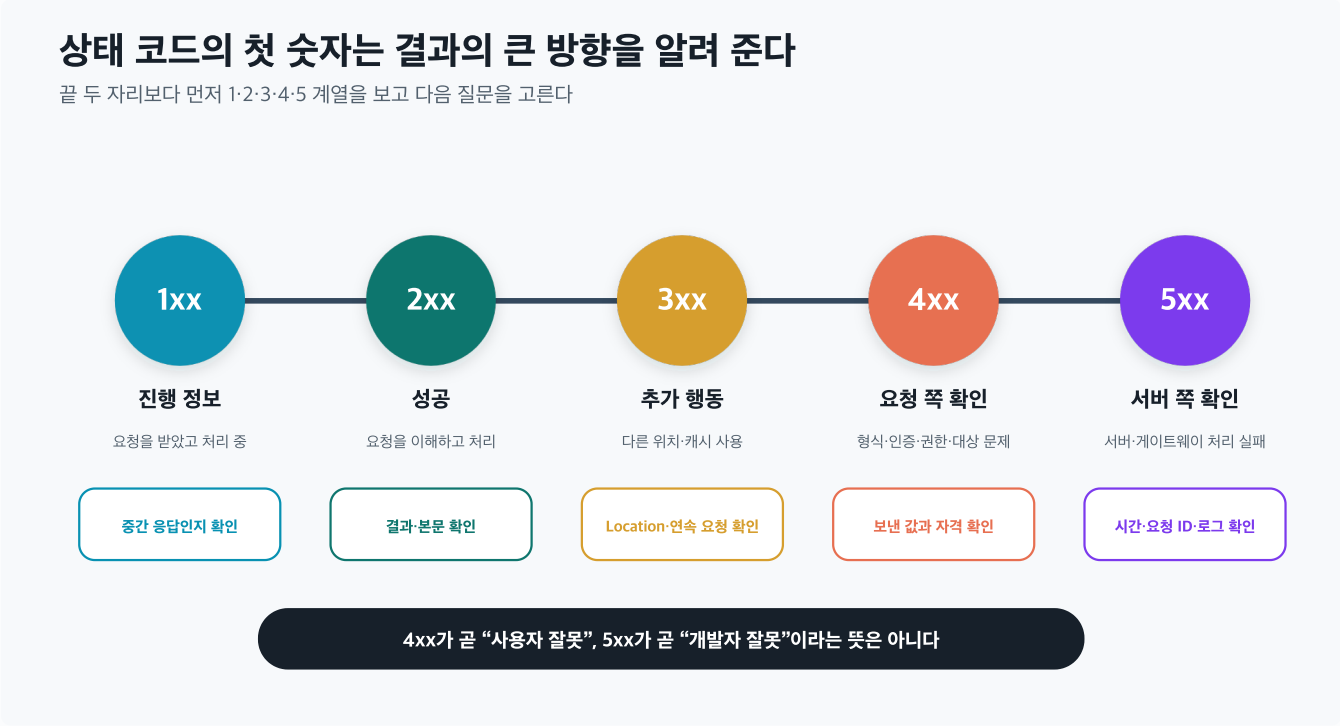


그림 6. 첫 숫자로 결과 계열을 고른 뒤 세 자리 전체 코드와 응답 상세를 확인합니다.


1xx
2xx
3
진행 정보
성공
추
요청을 받았고 처리 중
요청을 이해하고 처리
다른 위
중간 응답인지 확인
결과·본문 확인
Location·
4xx가 곧 “사용자 잘못”, 5xx가 곧

h 행동

치·캐시 사용

요청 쪽 확인

형식·인증·권한·대상 문제

서버 쪽 확인

서버·게이트웨이 처리 실패

연속 요청 확인

보낸 값과 자격 확인

시간·요청 ID·로그 확인

: "개발자 잘못"이라는 뜻은 아니다

계열	표준의 큰 의미	첫 질문
1xx	정보를 알리는 중간 응답	최종 응답이 뒤에 오는가
2xx	요청을 성공적으로 받거나 이해하고 처리	어떤 결과가 생성·반환됐는가
3xx	요청 완료에 추가 행동 필요	이동 주소·캐시·다음 요청은 무엇인가
4xx	요청을 처리할 수 없는 조건	형식·인증·권한·대상·호출량 중 무엇인가
5xx	서버가 유효해 보이는 요청 처리에 실패	어느 서버·게이트웨이·뒷단에서 실패했는가

오해하지 않기

- 2xx 라고 업무 결과가 항상 맞다는 뜻은 아님: 응답 본문의 업무 상태도 확인
- 4xx 라고 사용자의 실수로 단정하지 않음: 클라이언트 코드·명세·토큰 발급·서버 정책 문제일 수 있음
- 5xx 라고 한 서버만의 문제로 단정하지 않음: 게이트웨이·외부 연계·데이터베이스 지연일 수 있음
- 상태 코드를 받지 못한 네트워크 오류도 있음: DNS·연결·인증서 단계에서 HTTP 응답 전 실패 가능

9. 자주 만나는 상태 코드를 다음 행동으로 읽습니다

그림 7. 코드는 암기 목록이 아니라 다음 확인 순서를 고르는 표지판입니다.

200	**OK** 결과 본문 확인	201	**Created** Location 확인
304	**Not Modified** 캐시 사용 확인	400	**Bad Request** 형식·필수값
404	**Not Found** 경로·자원·공개 여부	409	**Conflict** 현재 상태·버전
500	**Internal Error** 요청 ID·서버 로그	502	**Bad Gateway** 앞단·뒷단 응답

204	**No Content**	본문 없음이 정상
302	**Found**	Location·다음 요청
401	**Unauthorized**	인증 자격
403	**Forbidden**	권한·정책
422	**Unprocessable**	값의 의미·검증
429	**Too Many**	Retry-After·호출량
503	**Unavailable**	점검·과부하·재시도
504	**Gateway Timeout**	뒷단 지연·시간

성공과 추가 행동

코드	뜻	다음 확인
`200 OK`	요청 성공	메서드에 맞는 결과 본문과 업무 상태
`201 Created`	새 자원 생성	`Location`·생성 식별자·중복 여부
`204 No Content`	성공했고 보낼 추가 본문 없음	화면·로컬 상태 갱신과 응답 헤더
`302 Found`	현재는 다른 URI에 있음	`Location`과 이어진 요청·메서드 변화
`304 Not Modified`	조건부 조회 결과 기존 표현 사용 가능	캐시가 어떤 응답을 재사용했는가

요청 쪽 조건 확인

코드	뜻	다음 확인
`400 Bad Request`	요청 형식 등을 처리할 수 없음	JSON·필수값·헤더·인코딩
`401 Unauthorized`	유효한 인증 자격이 없음	토큰 존재·만료·발급 대상·`WWW-Authenticate`
`403 Forbidden`	서버가 요청을 이해했으나 허용하지 않음	사용자·역할·자원 정책·망 정책
`404 Not Found`	자원을 찾지 못했거나 존재 공개를 원하지 않음	경로·식별자·환경·배포·권한 은폐 정책
`409 Conflict`	현재 자원 상태와 충돌	버전·중복·업무 상태·재조회
`422 Unprocessable Content`	형식은 읽었지만 의미 규칙을 처리할 수 없음	필드별 검증·업무 규칙·허용값
`429 Too Many Requests`	일정 시간에 너무 많은 요청	`Retry-After`·호출량·재시도 간격

서버와 연계 확인

코드	뜻	다음 확인
`500 Internal Server Error`	더 구체적인 5xx를 고르기 어려운 서버 오류	시간·요청 ID·서버 로그·직전 변경
`502 Bad Gateway`	게이트웨이가 뒷단에서 유효한 응답을 받지 못함	앞단·뒷단 주소·응답·연결
`503 Service Unavailable`	서버가 일시적으로 요청을 처리할 준비가 안 됨	점검·과부하·용량·`Retry-After`

코드	뜻	다음 확인
`504 Gateway Timeout`	게이트웨이가 뒷단 응답을 제시간에 못 받음	뒷단 처리 시간·시간 제한·외부 연계

구분 연습 1: `401` 과 `403`

`401` 은 유효한 인증 자격이 부족한 상황에 가깝고, `403` 은 자격을 알고 있더라도 허용하지 않는 상황에 가깝습니다. 실제 서비스는 보안을 위해 다른 코드를 선택할 수도 있으므로 응답 본문과 인증 정책을 함께 봅니다.

구분 연습 2: `400` 과 `422`

`400` 은 요청 문법·형식 등 넓은 요청 문제에 사용됩니다. `422` 는 요청 형식을 읽었지만 값의 의미나 업무 규칙 때문에 처리할 수 없을 때 사용됩니다. 서비스가 모든 검증 오류를 `400` 으로 통일하는 경우도 있으므로 명세가 우선입니다.

구분 연습 3: `502` 와 `504`

둘 다 게이트웨이 뒤의 서버와 관련될 수 있습니다. `502` 는 유효하지 않은 응답을 받았다는 쪽, `504` 는 제시간에 응답을 받지 못했다는 쪽에 초점을 둡니다.

10. Network에서 한 요청을 여섯 칸으로 읽습니다

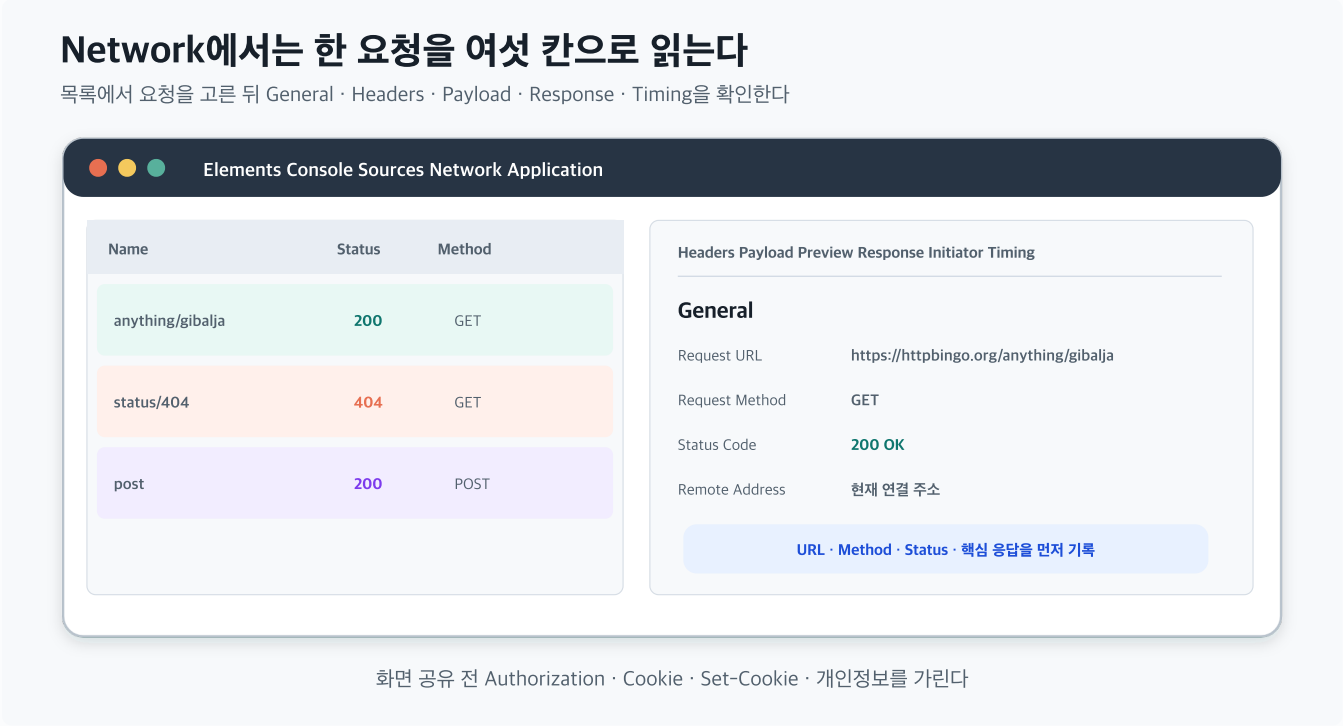
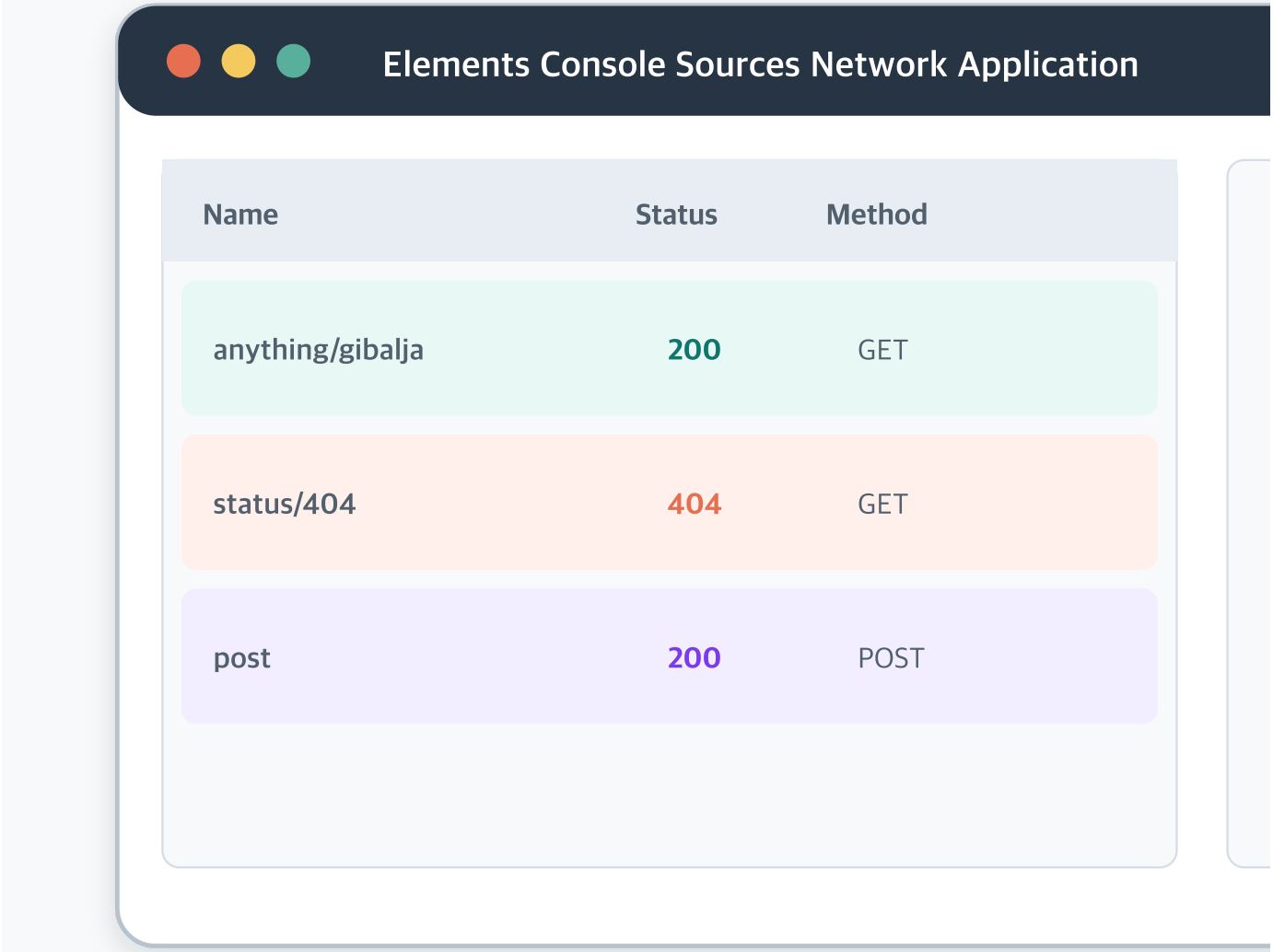


그림 8. 요청 목록에서 대상을 선택한 뒤 URL·메서드·상태·헤더·입력·응답·시간을 확인합니다.


Elements Console Sources Network Application
Name
Status
Method
anything/gibalja
200
GET
status/404
404
GET
post
200
POST

Headers Payload Preview Response Initiator Timing

General

Request URL	https://httpbingo.org/anything/gibalja
Request Method	GET
Status Code	200 OK
Remote Address	현재 연결 주소

URL · Method · Status · 핵심 응답을 먼저 기록

위치	보는 것	기록할 최소 정보
요청 목록	Name·Status·Method·Type·Time	실패 요청 한 줄
General	Request URL·Method·Status	전체 URL·메서드·상태 코드
Request Headers	요청 조건	`Content-Type`·인증 유무·추적값
Payload	질의·폼·요청 본문	민감정보를 가린 입력 구조
Response Headers	결과 조건	`Content-Type` · `Location` · `Retry-After` ·요청 ID
Response	결과·오류 본문	오류 코드·메시지·필드별 상세
Timing	대기·다운로드 시간	어느 구간이 오래 걸렸는가

Status가 `(failed)`인 경우

HTTP 상태 코드가 아니라 브라우저 오류가 표시될 수 있습니다. 이때는 HTTP 응답을 받기 전 DNS·연결·인증서·브라우저 보안 정책 단계에서 멈췄는지 M02-02의 오류 지도로 돌아갑니다.

`Provisional headers are shown`인 경우

캐시에서 처리됐거나 요청이 실제 네트워크로 나가지 않았거나 보안상 전체 헤더를 표시하지 못한 경우가 있습니다. “실제로 전송된 최종 헤더”라고 단정하지 않고 캐시·오류·요청 상태를 함께 확인합니다.

11. 오류를 재현 가능한 증거 묶음으로 바꿉니다

그림 9. 개발자가 같은 요청을 찾고 비교할 수 있도록 시간·환경·요청·응답·추적 정보를 묶습니다.

증상

사용자가 본 결과

요청

시간 · URL · 메서드

상태 ·

좋은 전

“15:20 운영 환경에서 POST /orders가 409,

응답

헤더 · 본문

연결

Timing · 반복 여부

추적

요청 ID · 환경

달 문장

. 요청 ID req-7f2a이며 두 번 재현됐습니다.”

좋지 않은 전달:

```
저장 안 돼요. 빨리 봐 주세요.
```

좋은 전달:

```
2026. 7. 15. 15:20 운영 환경에서 주문 저장을 눌렀습니다.
POST https://service.example.com/orders가 409를 반환했습니다.
응답의 errorCode는 ORDER_STATE_CONFLICT이고 요청 ID는 req-7f2a입니다.
새로 고친 뒤 두 번 재현됐고, 같은 계정의 조회 GET은 200입니다.
```

증거 7종 세트

1. 발생 시각과 표준시간대
2. 개발·시험·운영 환경
3. 사용자 행동과 기대 결과
4. 전체 URL에서 민감한 질의를 가린 값
5. 요청 메서드와 상태 코드
6. 오류 본문과 추적 식별자
7. 재현 횟수·직전 변경·Timing

> 증거를 많이 모으는 것과 민감정보를 많이 공유하는 것은 다릅니다. 토큰·쿠키·개인정보·내부 IP·전체 응답 데이터는 필요한 범위만 가려서 전달합니다.

12. 실습: 다섯 종류의 HTTP 대화 관찰하기

자세한 절차는 L02-03 실습 문서에 있습니다.

실습 A. GET 요청

```
https://httpbingo.org/anything/gibalja?topic=http
```

`General`, `Request Headers`, `Payload`, `Response Headers`, `Response`에서 메서드·질의·상태·본문을 기록합니다.

실습 B. 폼 POST 요청

```
https://httpbingo.org/forms/post
```

가상 정보만 입력해 제출하고 `post` 요청의 메서드와 Form Data를 확인합니다.

실습 C. 리다이렉션

```
https://httpbingo.org/redirect/1
```

Preserve log를 켜고 `302`와 이어지는 `200` 요청, `Location` 헤더를 확인합니다.

실습 D. 같은 화면, 다른 상태

```
https://httpbingo.org/status/404
https://httpbingo.org/status/503
```

본문이 비어 보여도 Network에는 상태 코드가 남는지 확인합니다.

완료 산출물

- GET 요청·응답 해부표 1개
- POST 요청·응답 해부표 1개
- `302 → 200` 이동 기록 1개
- `404`와 `503` 판단표 1개

13. 인쇄용 HTTP 분석 학습지

A. 요청 해부표

항목	기록	내가 이해한 역할
발생 시각·환경		
전체 URL		
메서드		
요청 대상 경로·질의		
중요 요청 헤더		
요청 본문·Payload		
민감정보 가림 여부		

B. 응답 해부표

항목	기록	다음 확인
상태 코드·계열		
`Content-Type`		
`Location` · `Retry-After`		
요청·추적 ID		
응답 본문·오류 코드		
Timing		

C. 상태 코드 판단

질문	기록
첫 숫자가 가리키는 계열은 무엇인가	
이 코드가 직접 말해 주는 사실은 무엇인가	
아직 알 수 없는 것은 무엇인가	
다음에 볼 헤더·본문·로그는 무엇인가	
누구에게 어떤 문장으로 전달할 것인가	

D. 요청·응답 직접 그리기

```
[클라이언트]
   └ ______  /_____________  HTTP/___
     헤더: _____________________________
     본문: _____________________________

                          [서버]
   ┌ HTTP/___  ______  _________________
   │  헤더: _____________________________
   └ 본문: _____________________________

다음 확인 행동: ___________________________________________
```

14. 셀프 테스트

문제 1 · 공통 구조

HTTP 요청과 응답이 공유하는 네 구역을 순서대로 쓰세요.

문제 2 · 요청 해부

다음 요청에서 메서드, 요청 대상, 본문 형식, 본문을 찾으세요.

```
POST /notes HTTP/1.1
Host: api.example.com
Content-Type: application/json

{"title":"HTTP"}
```

문제 3 · 응답 해부

다음 응답에서 상태 코드, 새 자원의 위치, 본문을 찾으세요.

```
HTTP/1.1 201 Created
Location: /notes/7
Content-Type: application/json

{"id":7}
```

문제 4 · 메서드

조회·새 작업 처리·전체 교체·부분 변경·연결 제거에 일반적으로 대응하는 메서드를 쓰세요.

문제 5 · 상태 계열

`103`, `204`, `304`, `404`, `504`를 각 상태 코드 계열과 연결하세요.

문제 6 · 성공 응답

`200`, `201`, `204`의 차이를 한 문장씩 설명하세요.

문제 7 · 인증과 권한

`401`과 `403`을 구분하고 각각 다음에 확인할 항목을 쓰세요.

문제 8 · 게이트웨이 오류

`502`와 `504`가 각각 무엇에 초점을 두는지 쓰세요.

문제 9 · 보안

Network 화면을 공유하기 전에 가려야 할 값 네 가지를 쓰세요.

문제 10 · 전달 문장

다음 정보를 사용해 개발자에게 전달할 한 문장을 만드세요.

```
시각: 15:20 KST
환경: 운영
메서드·경로: POST /orders
상태: 409
요청 ID: req-7f2a
재현: 2회
```

15. 정답과 해설

문제 1

`시작줄 → 헤더 → 빈 줄 → 선택적 본문` 입니다.

문제 2

- 메서드: `POST`
- 요청 대상: `/notes`
- 본문 형식: `application/json`
- 본문: `{"title":"HTTP"}`

문제 3

- 상태 코드: `201`
- 새 자원의 위치: `/notes/7`
- 본문: `{"id":7}`

문제 4

- 조회: `GET`
- 새 작업 처리: `POST`
- 전체 교체: `PUT`
- 부분 변경: `PATCH`
- 연결 제거: `DELETE`

서비스의 실제 의미와 허용 범위는 명세에서 다시 확인합니다.

문제 5

- `103` : 1xx 정보
- `204` : 2xx 성공
- `304` : 3xx 추가 행동·캐시
- `404` : 4xx 요청 조건 확인

- `504` : 5xx 서버·게이트웨이 확인

문제 6

- `200` : 메서드에 따른 일반적인 성공 결과
- `201` : 하나 이상의 새 자원이 생성됨
- `204` : 성공했지만 보낼 추가 본문이 없음

문제 7

`401` 은 유효한 인증 자격이 없음을, `403` 은 서버가 요청을 허용하지 않음을 가리킵니다. `401` 에서는 토큰·세션·발급 대상을, `403` 에서는 역할·자원 권한·정책을 우선 확인합니다.

문제 8

`502` 는 게이트웨이가 뒷단에서 유효한 응답을 받지 못한 상황, `504` 는 뒷단 응답을 제시간에 받지 못한 상황에 초점을 둡니다.

문제 9

예시 답: `Authorization` , `Cookie` , `Set-Cookie` , 접근 토큰, 개인정보, 내부 IP 중 네 가지입니다.

문제 10

예시 답:

> 15:20 KST 운영 환경에서 `POST /orders` 가 `409` 를 반환했고, 요청 ID는 `req-7f2a` 이며 같은 조건에서 두 번 재현됐습니다.

16. 한 장 요약

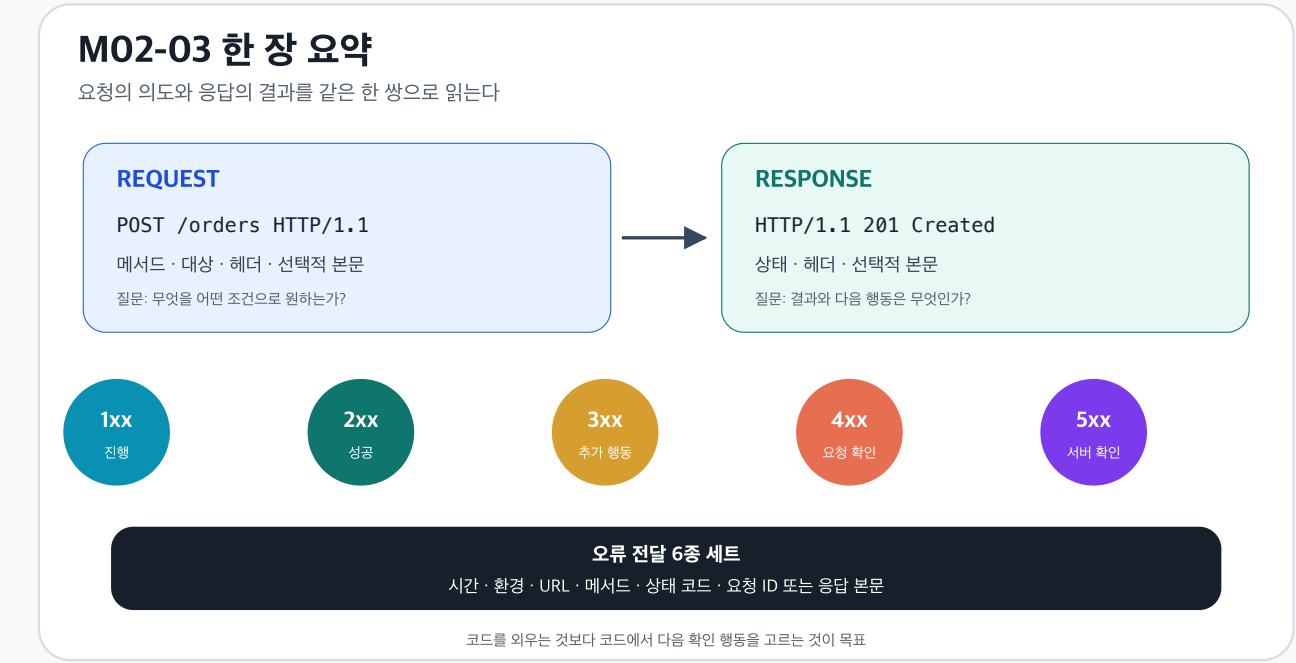


그림 10. 요청의 의도와 응답의 결과를 한 쌍으로 읽고, 상태 코드를 다음 확인 행동으로 바꾸세요.

REQUEST

`POST /orders HTTP/1.1`

메서드 · 대상 · 헤더 · 선택적 본문

질문: 무엇을 어떤 조건으로 원하는가?

1xx
진행

2xx
성공

3xx
추가 행동

오류 전달

시간 · 환경 · URL · 메서드 · 상

RESPONSE

`HTTP/1.1 201 Created`

상태 · 헤더 · 선택적 본문

질문: 결과와 다음 행동은 무엇인가?

4xx
요청 확인

5xx
서버 확인

6종 세트

태 코드 · 요청 ID 또는 응답 본문

17. 다음 학습과 출처

17.1. 다음 매뉴얼

M02-04 「브라우저 개발자 도구로 API 찾기」에서 여러 요청 중 실제 데이터 요청을 찾고, API 호출을 분석 문서로 정리합니다.

17.2. 출처와 확인일

- MDN, HTTP messages, 2026. 7. 15. 확인
- RFC Editor, RFC 9110: HTTP Semantics, 2026. 7. 15. 확인
- IANA, HTTP Status Code Registry, 2026. 7. 15. 확인
- MDN, HTTP response status codes, 2026. 7. 15. 확인
- Chrome for Developers, Network features reference, 2026. 7. 15. 확인

변경 이력: v0.1.0 · 2026. 7. 15. · 파일럿 초안, 도표 10개, 실습, 셀프 테스트 통합