Engineer

HTTP

⇄ ウェブを成り立たせる要求と応答の規則

どのようなものか

HTTP（Hypertext Transfer Protocol）は、ブラウザー、アプリ、スクリプトが、サーバーに何かを求めて返答を受け取るための規則です。その会話を暗号化したTLS接続の中で行うと、同じプロトコルをHTTPSと呼びます。やり取りは、二つの部分からなります。クライアントは、`GET` や `POST` などのメソッド、求める対象、ヘッダーフィールドの一覧、ときには本文を、要求として送ります。サーバーは、三桁の状態コード、自分側のヘッダーフィールド、通常は本文で応答します。ページの読み込み、画像の取得、プログラムから別のプログラムへの呼び出しは、すべて、このやり取りの繰り返しです。

```
GET /pricing/ HTTP/1.1
Host: example.com
Accept: text/html

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Cache-Control: max-age=600
```

仕様は、これをステートレスと呼びます。各要求には、サーバーが理解するために必要なものをすべて含める必要があり、プロトコルは、やり取りの間の記憶を持ちません。ログイン、買い物かご、セッションは、その上に、通常はCookieを使って組み立てます。サーバーが一度設定したヘッダーフィールドを、クライアントが後の要求のたびに送り返します。

語彙は少なく、正確です。`GET` と `HEAD` は安全とされます。表現を求め、何も変更しないと期待される意味です。`PUT` と `DELETE` は冪等です。同じものを二回送っても、一回送った場合とサーバーの状態が同じになります。`POST` は、どちらも約束しません。状態コードは先頭の桁で分類され、2xxが成功、3xxが転送、4xxが要求側の問題、5xxがサーバー側の問題です。クライアントは、知らないコードでも、おおよそ何が起きたかを把握できます。

通信上の形式が二回置き換わっても、その意味は変わっていません。HTTP/1.1は、TCP接続上でメッセージをテキストの行として書き、各接続では、一つずつ順番に応答を返します。HTTP/2は、同じメソッドやフィールドを保ちつつ、バイナリーのフレームへ収め、一つの接続で多数の要求を交互に流します。HTTP/3は、同じ意味をQUIC上で実現します。QUICはUDP上に構築された転送方式で、独立したストリームの間では、TCPのトランスポート層にある先頭ブロッキング（head-of-line blocking）を避けられます。2022年からIETFは、意味を一つの文書、各版の通信形式を別々の文書で公開しています。そのため、「404は何を意味するか」と「バイトをどう並べるか」には、今は別々の答えがあります。

したがってHTTPとHTTPSの比較は、二つのプロトコルの対決ではありません。HTTPSは、下にTLSを敷いたHTTPで、やり取りを暗号化し、サーバー名を証明します。WebSocketは、HTTP要求として始まり、その後、接続を双方向の通信路へ切り替えます。gRPCは、HTTP/2上で動く、遠隔の関数を呼び出すための規約です。

概念を判断に生かす

「HTTPとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 Wi-Fi、ファイアウォール、リモートアクセスを整え、どこからでも働け、権限のある人だけが接続できるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Ubuntu Linux VPSインスタンス30台を管理 — 30 復旧計画とともに管理したLinuxサーバー

Ubuntu Linux VPSインスタンス30台を管理 お問い合わせ →

なぜ重要なのか

インターネット越しに別のシステムと話すシステムのほぼすべてが、すでにHTTPを使っています。そのため、重要な決定が、使うかどうかになることはほとんどありません。定義された機能を使うか、単なる通信路として使って残りを自作するかです。

十分に活用すると効果があるのは、長い処理経路を構成するソフトウェアが、すでにその語彙を理解するからです。クライアントライブラリーは、時間切れになった `GET` を再試行する場合があります。 `POST` は再試行してはならず、途中のプロキシーも同じです。キャッシュは、 `max-age=600` と記された応答を十分間保存し、サーバーへ問い合わせずに、次の読者へ渡せます。監視は、指示されなくて

も5xxの応答をエラーと数えます。すべての呼び出しを、一つのアドレスへの `POST` で送り、失敗を200の応答内で報告するAPIは、それらをすべて捨て、どのクライアントにも、そのAPI独自の規則を覚えさせます。

このモデルにも限界があります。要求と応答は、欲しいものが分かっているクライアントには適しますが、新しい情報を送り出したいサーバーには、あまり適しません。そのため、リアルタイムの更新には、通常、WebSocket、サーバー送信イベント（server-sent events）、ポーリングを使います。

公開インターネット上の平文HTTPは、別の問いで、答えは短いものです。ブラウザーが安全でないと表示し、経路上の誰もが読み取って変更でき、証明書は無料なので、公開通信にはHTTPSを使います。HTTP/1.1、HTTP/2、HTTP/3の選択は、ほぼサーバーか、その前段のエッジの設定です。どの要求も意味が同じであるため、アプリケーションコードは変わりません。

実際の活用例

このサイトで最も明確な例は、HTTP自身の構造に沿ったフレームワークへ移したAPIです。HTTP APIをFiberからHuma v2へ移行しました。HumaはFiberの上で動き、各エンドポイントを入出力の構造体で宣言し、手書きの検査ではなく、型の宣言から検証を生成します。同じ宣言からOpenAPI記述を生成し、CIの検査で、サーバーが登録する経路と、記述に掲載する経路を比較します。649パス、760操作で一致率は100%です。誰も記述していない経路があればビルドが失敗するため、それを保っています。Fiberを直接使うまま残したエンドポイントは、ちょうど二つです。Humaの要求と応答のモデルに適さない、WebSocketのものです。

サイト自身は、コンテンツネゴシエーションを使います。すべてのページをHTMLとMarkdownで公開し、Markdownを得る一つの方法は、`Accept: text/markdown` を付けた要求を送り、サーバーに選ばせることです。同じページのonionアドレスが平文の `http://` なのは、意図的です。そのアドレスにサービスの公開鍵が符号化されているため、最初のHTTP要求が流れるまでに、Torがすでに通信路を認証し、暗号化しています。

HTTPは、インフラコードの道具としても登場します。二台目のサーバーを用意したとき、両事業者のファイアウォールを、それぞれのREST APIへ直接要求して設定しました。事業者別のコレクションの代わりに、汎用HTTPタスクを使いました。依存関係と、そのバージョンのずれをなくした一方、各要求の形は手で書く必要が生じました。事例は、さらにもう一つの負担を挙げます。汎用HTTPタ

スクはチェックモードへの対応を宣言していないため、マシンを作るプレイの試行実行では、その中のすべてのタスクが省略されます。そのため、予行演習は、代わりにファイアウォールのプレイで行います。

問題が起きるところ

HTTPの不具合の大半は、その語彙のある言葉を、途中のほかのソフトウェアが受け取る意味と違う意味で使うことから起きます。

成功として届くエラー

APIが200を返し、本文へ `"error": "card declined"` を入れます。キャッシュが保存する場合があります。ダッシュボードは成功と数え、クライアントの再試行処理は動きません。適切な4xxか5xxの状態コードを送り、本文で問題を説明します。RFC 9457は、そのためのJSONの形を定め、 `application/problem+json` として提供します。

何かを削除するリンク

メール内の登録解除や削除のリンクなど、 `GET` 要求が状態を変更します。クローラー、チャットアプリのリンクプレビュー、セキュリティスキャナー、ブラウザーの先読みは、どれも自らリンクをたどり、それぞれがその処理を実行させる可能性があります。データを変える処理には、 `POST` 、 `PUT` 、 `DELETE` のいずれかを使い、リンクが必要なら、確認ページを設けます。

二回引き落とされる支払い

クライアントが `POST` を送り、応答が届く前に接続が切れて、再試行します。サーバーは、すでに一回目を処理していました。HTTPは、クライアントに、それらの場合を区別する方法を与えないため、APIが対処します。ヘッダーで冪等性キーを受け取り、同じキーには保存した結果を返すか、クライアントが新しいリソースの識別子を選び、 `PUT` で作成するようにします。

GETへ変わるフォーム

サーバーが `POST` へ301か302を返し、別の場所へ転送します。歴史的な理由から、仕様は、この二つのコードで、ユーザーエージェントがメソッドを `GET` へ変えることを認め、多くがそうします。そのため、途中で本文が失われます。転送後もメソッドを保つ必要があるときは、307か308を使います。

キャッシュから届く、違う版

一つのURLが、要求ヘッダーフィールドに応じて、HTMLかMarkdown、英語かデンマーク語を提供します。何も伝えられていないキャッシュは、最初に見た版を保存し、次の読者へ渡します。応答は、たとえば `Vary: Accept` のように、依存するフィールドを挙げる必要があります。それにより、キャッシュは、URLごとに一つではなく、版ごとに一つを保存します。

参考資料

- RFC 9110：HTTPの意味 — すべての版に共通して、一度だけ定義するメソッド、状態コード、ヘッダーフィールドです。
- RFC 9111：HTTPキャッシュ — 保存した応答を再利用できる条件と、`Cache-Control` がキャッシュへ伝える内容です。
- RFC 9112：HTTP/1.1 — テキストの通信形式、接続の扱い、メッセージの区切り方です。
- RFC 9113：HTTP/2 — バイナリーのフレームと、一つの接続上の多数のストリームです。
- RFC 9114：HTTP/3 — 同じ意味をQUICのストリームへ対応させたものです。
- RFC 9457：HTTP APIの問題詳細 — 問題の内容を記述する、標準のJSON本文です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「HTTPとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/http/