Das HTTP Protokoll

Was ist HTTP?

HTTP (Hypertext Transfer Protocol) ist ein zustandsloses (state-less) Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz. Es ermöglicht den Austausch von Hypermedia-Dokumenten im Web, typischerweise zwischen einem Browser und einem Server, damit Menschen sie lesen können. Hier sind die wichtigsten Punkte:

Was ist ein Protokoll?

Ein **Protokoll** ist eine festgelegte Abfolge von Regeln und Verfahren, die es ermöglicht, dass zwei oder mehr Systeme miteinander kommunizieren können. Es definiert, wie Daten ausgetauscht werden, welche Nachrichtenformate verwendet werden und wie Fehler behandelt werden.

Sockets, Ports und Socket-Pair

HTTP verwendet für den Datentransport ausschließlich TCP/IP. Hierzu werden 2 Sockets benötigt, ein Server-Socket im `listen` Modus und ein Client-Socket im `connect` Modus. Standard Ports für den Server-Socket sind Port 80 (`http`) sowie Port 443 (`https`), jedoch kann ein HTTP-Server auch auf jedem beliebigen anderen Port lauschen, wie zB. der Live-Server auf Port 5500.

Socket: Ist die Kombination aus IP-Adresse und Port-Nummer.

Socket-Pair: Ist die Verbindung zweier Sockets, also eine "stehende" (TCP) Verbindung eines Client-Sockets mit einem Server-Socket.

Was heißt idempotent?

"Eine idempotente Funktion liefert dasselbe Ergebnis, selbst wenn sie mehrmals angewendet wird." Im Kontext von HTTP bedeutet das, dass eine idempotente Anfrage mehrmals ausgeführt werden kann, ohne dass sich das Ergebnis (in der Applikation / Datenbank) ändert. Das ist wichtig, um sicherzustellen, dass die Anfrage nicht unerwünschte Nebenwirkungen hat.

HTTP-Verben (Client-Methoden)

Bei HTTP verwendet der Client verschiedene Verben (Befehle, Methoden), um Aktionen auf Ressourcen auszuführen. Die **Verben** verwendet der **client**. Der **Server** antwortet mit **Status Codes** (s.u.). Es gibt teilweise Verwandschaft mit CRUD-Operationen (Create, Read, Update, Delete) in Datenbanken:

- **GET**: Abrufen von Daten von einem Server. (CRUD: Read (==select; idempotent))
- **POST**: Senden von Daten an einen Server (z. B. Formulardaten) (CRUD: Create (==insert; NICHT idempotent)).
- **PATCH**: Teilweise Aktualisierung einer Ressource. (CRUD: Update (==update; idempotent)).
- **DELETE**: Löschen einer Ressource auf dem Server. (CRUD: Delete (==delete; idempotent)
- **HEAD**: Ähnlich wie GET, aber ohne den eigentlichen Inhalt abzurufen.
- **PUT**: Aktualisieren einer vorhandenen Ressource auf dem Server.

URL

RFC 3886: https://www.rfc-editor.org/info/rfc3986 Siehe https://developer.mozilla.org/en-US/docs/Web/API/URL.

"Uniform Ressource Locator", zb. https://www.orf.at/ oder http://localhost:8080/app oder auch https://anonymous:flabada@developer.mozilla.org:4711/en-US/docs/Web/API/URL/password#my-item.

Namen der Bestandteile im letzten Beispiel:

- Protokoll (https)

- username (*optional*: anonymous)
- passwort (*optional*: flabada)
- port (*optional, aber notwendig wenn nicht default*: 4711)
- path (*optional, default to `/`*: /en-US/docs/Web/API/URL/password)
- search-params (*optional: hier: anchor*: my-item)

Search-Parameter werden nach einem `?` mitgesendet, z.B. "?q=mysearch&display=true" und werden von backend-software extra bereitgestellt.

Weiters sind in URLs viele Zeichen nicht erlaubt, z.B. Leerzeichen uvm., deswegen muss eine URL `url-encoded` sein. Siehe RFC oder entsprechende Funktionen der Programmierspachen z.B. `java.net.URLEncoder`.

HTTP Statuscodes (Server-Antwort)

HTTP-Statuscodes sind 3-stellige Zahlen am Beginn der Server-Antwort, die den Status einer Anfrage anzeigen.

- **1xx**: Informational - Die Anfrage wurde erhalten und verarbeitet. (sehr unüblich)
- **2xx**: Erfolgreich - Die Anfrage wurde erfolgreich verarbeitet.
- **3xx**: Umleitung - Der Client muss eine weitere Aktion ausführen, um die Anfrage abzuschließen.
- **4xx**: Client-Fehler - Die Anfrage konnte nicht verarbeitet werden, da der Client einen Fehler gemacht hat.
- **5xx**: Server-Fehler - Der Server konnte die Anfrage nicht verarbeiten, weil ein Serverfehler aufgetreten ist.

Beispiele:

- **200 OK**: Die Anfrage war erfolgreich.
- **301 Moved Permanently**: Die angeforderte Ressource wurde dauerhaft an eine andere URL verschoben.
- **302 Found**: Die angeforderte Ressource wurde vorübergehend an eine andere URL verschoben.
- **303 See Other**: Der Client sollte eine andere URL verwenden, um die Anfrage abzuschließen.
- **401 Unauthorized**: Der Client muss sich authentifizieren, um Zugriff zu erhalten.
- **403 Forbidden**: Der Client hat keine Berechtigung, auf die angeforderte Ressource zuzugreifen.
- **404 Not Found**: Die angeforderte Ressource wurde nicht gefunden.
- **500 Internal Server Error**: Ein allgemeiner Serverfehler ist aufgetreten.
- **503 Service Unavailable**: Der Server ist vorübergehend nicht verfügbar.

HTTP-Header

HTTP-Header sind Zusatzinformationen (Metadaten), die Informationen über die Anfrage und über die Antwort enthalten. Sie können mit F-12 in den Developertools eingesehen werden. Am Ende der Header kommt zwei Mal hintereinander ein `newline`, daran erkennt die Gegenstelle, dass ab es jetzt mit der tatsächlichen `payload` weitergeht.

typische Client - Header (vom Browser gesetzt):

- Referrer
- Host
- Cookie
- Content-Type

typische Server - Header:

- date
- content-type
- cache-control

Content-Type und **MIME-Typen**

MIME-Typen (Multipurpose Internet Mail Extensions) identifizieren den Medientyp einer Ressource. Der Content-Type-Header verwendet MIME-Typen, um den Medientyp anzugeben. Server UND Client verwenden diesen Header, um dem Partner mitzuteilen, wie er die Antwort interpretieren soll. Beispiele:

- `Content-Type: text/html` für HTML-Dokumente.
- `Content-Type: application/json` für JSON-Daten.

- `Content-Type: application/x-www-form-urlencoded` für Formulardaten (default bei `<form method="POST">`)

Auch noch sehr häufig sind:

- text/plain: Dies ist der Standardtyp für Textdateien. Textdateien sollten menschenlesbar sein und dürfen keine binären Daten enthalten.
- image/png: Dieser MIME-Typ wird für Portable Network Graphics (PNG)-Bilder verwendet.
- application/pdf: Adobe Portable Document Format (PDF)-Dateien haben diesen MIME-Typ.
- video/mp4: MP4-Videos verwenden diesen MIME-Typ.
- audio/mpeg: MP3-Audiodateien verwenden diesen MIME-Typ.

Cookies

- Cookies sind kleine Textdateien, die vom Server an den Client gesendet werden und auf dem Client gespeichert werden.
- Sie werden verwendet, um Informationen zwischen Anfragen zu speichern (z. B. Sitzungsdaten, Benutzerpräferenzen).
- Fungieren als Mechanismus zur Zustandsspeicherung (remember: **http ist stateless**).
- Der Server kann Cookies mit dem `Set-Cookie`-Header senden, und der Client sendet sie mit dem `Cookie`-Header zurück.

Sicheres HTTP (HTTPS)

- HTTPS ist eine verschlüsselte Version von HTTP, die die Kommunikation zwischen Client und Server absichert.
- Es verwendet SSL/TLS-Zertifikate, um Daten zu verschlüsseln und die Integrität sicherzustellen.
- HTTPS schützt vor Abhören, Manipulation und Datendiebstahl.