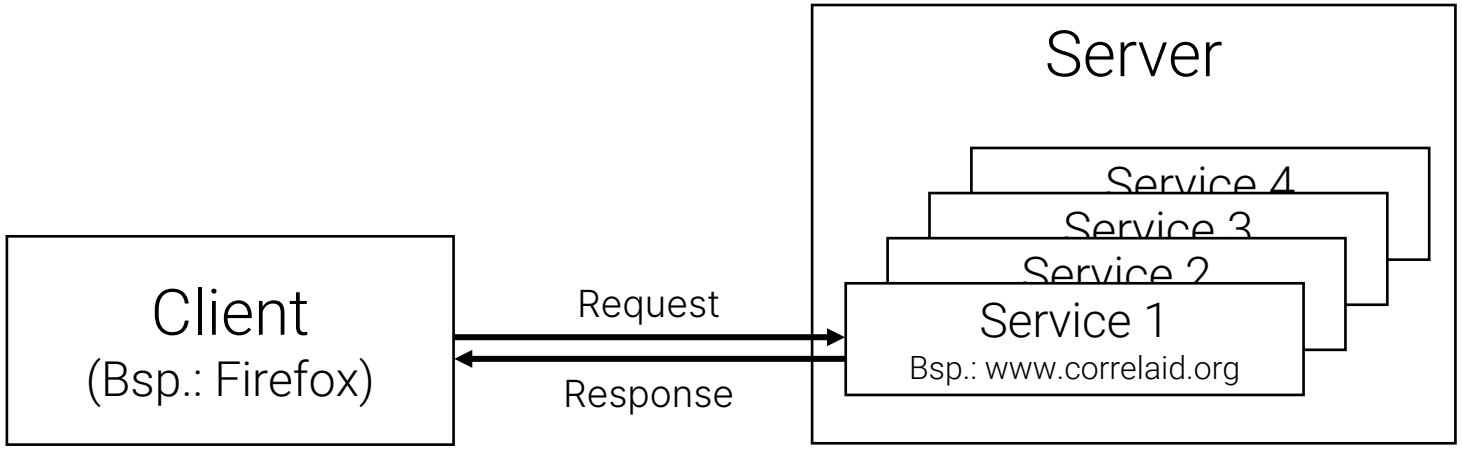
Datenzugriff im World Wide Web

Inhalt

1. Client-Server-Modell
2. Hypertext Transfer Protocol
3. Authenthentifizierung
4. Praktischer Teil 1
5. Praktischer Teil 2

Client-Server-Modell

Client-Server-Modell



Hypertext Transfer Protocol

Hypertext Transfer Protocol

Hypertext Transfer Protocol (HTTP)

- HTTP nutzt standartmäßig Port 80 zur Übertragung von Informationen.
  Bsp.: correlaid.org:80 oder http://correlaid.org
- HTTP ist ein zustandloses Protokoll.

Hypertext Transfer Protocol Secure (HTTPS)

- HTTPS ermöglicht eine verschlüsselte HTTP-Verbindung.
- HTTPS nutzt standartmäßig Port 443 zur Übertragung von Informationen.
  Bsp.: correlaid.org:443 oder https://correlaid.org

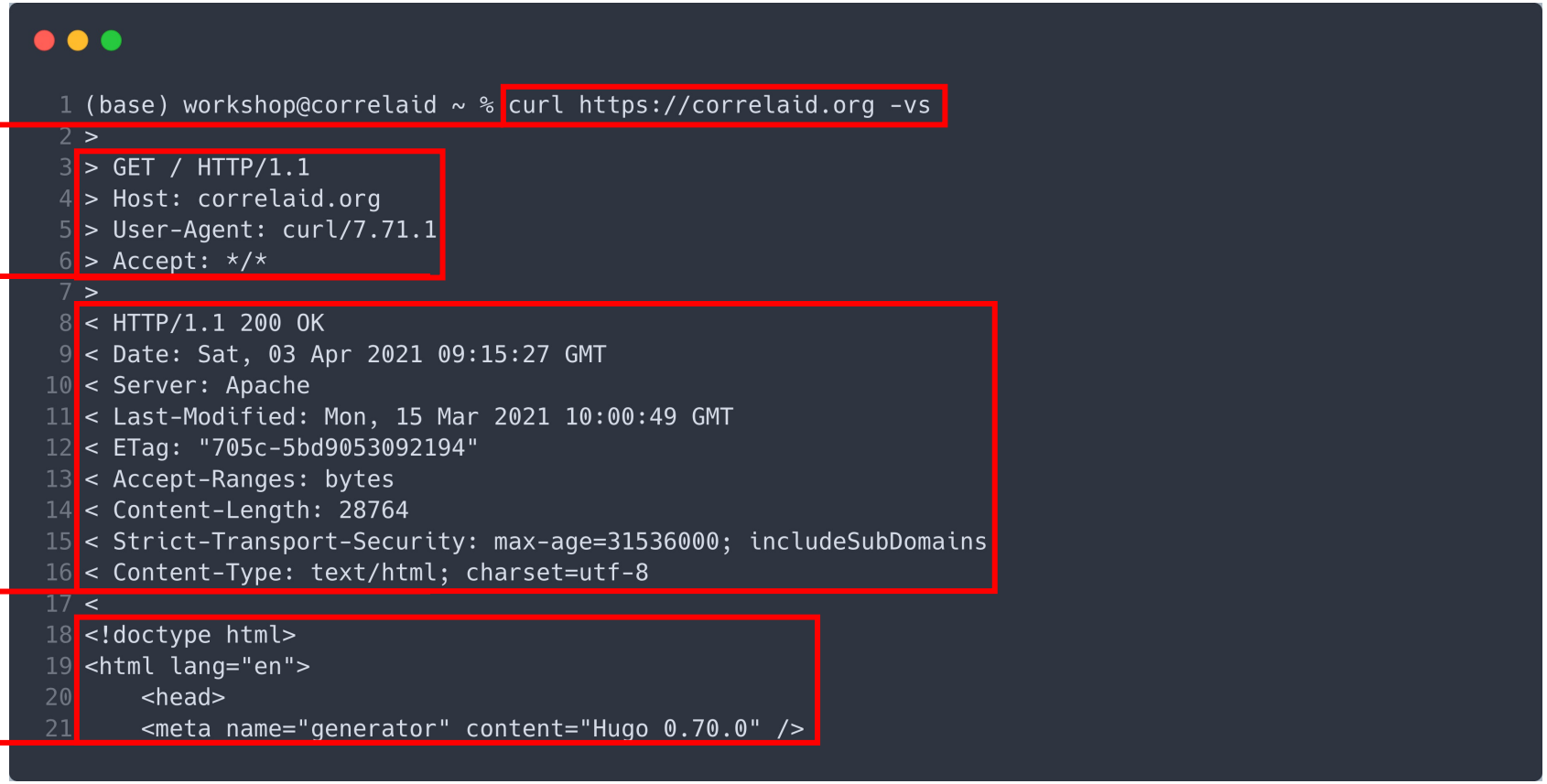
HTTP Anfrage

curl Command

Request Headers

Response Headers

Response Body



```
(base) workshop@correlaid ~ % curl https://correlaid.org -vs
>
> GET / HTTP/1.1
> Host: correlaid.org
> User-Agent: curl/7.71.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 03 Apr 2021 09:15:27 GMT
< Server: Apache
< Last-Modified: Mon, 15 Mar 2021 10:00:49 GMT
< ETag: "705c-5bd9053092194"
< Accept-Ranges: bytes
< Content-Length: 28764
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< Content-Type: text/html; charset=utf-8
<
<!doctype html>
<html lang="en">
    <head>
    <meta name="generator" content="Hugo 0.70.0" />
```

Request Headers

Pfad

HTTP Methode

Protokoll Version

Accept Header (MIME-Type)

User-Agent Header

```
(base) workshop@correlaid ~ % curl https://correlaid.org -vs
>
> GET / HTTP/1.1
> Host: correlaid.org
> User-Agent: curl/7.71.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 03 Apr 2021 09:15:27 GMT
< Server: Apache
< Last-Modified: Mon, 15 Mar 2021 10:00:49 GMT
< ETag: "705c-5bd9053092194"
< Accept-Ranges: bytes
< Content-Length: 28764
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< Content-Type: text/html; charset=utf-8
<
<!doctype html>
<html lang="en">
    <head>
    <meta name="generator" content="Hugo 0.70.0" />
```

Response Headers

Status Code

Protokoll Version

Status Nachricht

Content-Type Header

```
(base) workshop@correlaid ~ % curl https://correlaid.org -vs
>
> GET / HTTP/1.1
> Host: correlaid.org
> User-Agent: curl/7.71.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 03 Apr 2021 09:15:27 GMT
< Server: Apache
< Last-Modified: Mon, 15 Mar 2021 10:00:49 GMT
< ETag: "705c-5bd9053092194"
< Accept-Ranges: bytes
< Content-Length: 28764
< Strict-Transport-Security: max-age=31536000; includeSubDomains
< Content-Type: text/html; charset=utf-8
<
<!doctype html>
<html lang="en">
    <head>
    <meta name="generator" content="Hugo 0.70.0" />
```

HTTP Methoden

Zentrale Methoden:
- GET: Daten lesen
- POST: Neue Daten erstellen
- PUT: Daten ersetzen
- PATCH: Daten akutalisieren
- DELETE: Daten löschen

CRUD-Pattern:
- Create → POST
- Read → GET
- (List) → GET
- Update → PUT/PATCH
- Delete → DELETE

Quelle: https://developer.mozilla.org/de/docs/Web/HTTP/Methods

Dateneingabe 1

Path Parameter
- sind Teil der URL;
- können für alle HTTP-Methoden genutzt werden;
- können gecached werden und werden normalerweise in den Access-Logs gespeichert;
- sollten niemals unverschlüsselte Passwörter oder sonstige sensible Informationen beinhalten.

Bsp.: https://wish.com/items/*computer-xyz*

Query Parameter
- sind Teil der URL;
- können für alle HTTP-Methoden genutzt werden;
- können gecached werden und werden normalerweise in den Access-Logs gespeichert;
- sollten niemals unverschlüsselte Passwörter oder sonstige sensible Informationen beinhalten.

Bsp.: https://wish.com/items?*item=computer-xyz*

Dateneingabe 2

Body Parameter

- sind nicht Teil der URL;
- sind nicht Teil der Spezifikation von GET und werden hauptsächlich mit POST verwendet;
- werden normalerweise nicht gecached auch nicht in den Access-Logs gespeichert;
- können genutzt werden, um Passwörter oder sonstige sensible Informationen zu übertragen.

Bsp.: https://wish.com/items

Status Codes

1xx: Informative Antworten
2xx: Erfolgreiche Antworten
3xx: Umleitungen
4xx: Client-Fehler
5xx: Server-Fehler

Quelle: https://developer.mozilla.org/de/docs/Web/HTTP/Status

Status Codes: 2xx

200 OK: Anfrage erfolgreich
201 CREATED: Anfrage erfolgreich und Content wurde erstellt
204 NO CONTENT: Anfrage erfolgreich aber es gibt keinen Inhalt

Quelle: https://developer.mozilla.org/de/docs/Web/HTTP/Status

Status Codes: 4xx

400 BAD REQUEST: Fehler in der Anfrage
401 UNAUTHORIZED: Anfrage muss erst authentifiziert werden
403 FORBIDDEN: Kein Zugriff
404 NOT FOUND: Der Inhalt ist nicht bekannt

Quelle: https://developer.mozilla.org/de/docs/Web/HTTP/Status

Status Codes: 5xx

500 INTERNAL SERVER ERROR: Fehler auf Serverseite
501 NOT IMPLEMENTED: HTTP Methode wird nicht unterstützt
502 BAD GATEWAY: Weiterleitung innerhalb des Servers funktioniert nicht
503 SERVICE UNAVAILABLE: Anfrage kann nicht bearbeitet werden
504 GATEWAY TIMEOUT: Weiterleitung antwortet nicht

Quelle: https://developer.mozilla.org/de/docs/Web/HTTP/Status

Authentifizierung

Authentication Schemas

- Basic access authentication (Basic Auth)
- Digest authentication
- Bearer authentication (Token Auth)

Quelle: https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

Übergabe

- Authorization headers
- Body
- (URL)

Authorize requests

All requests to our API must be authorized, so we know who plays with our data. To keep URLs simple and clean, the key should be sent as an X-Authorization header attached to your HTTP request.

EXAMPLE

```
GET /{endpoint} HTTP/1.1
Host: http://api.zeit.de
X-Authorization: {api_key}
```

The key may also be sent as a query parameter, but a header is preferred.

EXAMPLE

```
GET /{endpoint}?api_key={api_key} HTTP/1.1
Host: http://api.zeit.de
```

Quelle: http://developer.zeit.de/quickstart/

Praktischer Teil 1

Praktischer Teil 2

Vielen Dank.