HTTP Server

- Einfache Implementation eines HTTP/1.1 Servers in Java
- Ohne Nutzung von Bibliotheken in der Laufzeitumgebung

Testen

Unter Windows in Powershell: `./gradlew.bat run`.

Dann sollte eine Beispielwebsite im Browser unter "http://127.0.0.1:8080" erreichbar sein.

Hinweise

Dies ist keine vollständige Implementation des HTTP/1.1 Protokolls, da einige Funktionen teilweise unvollständig sind oder komplett fehlen. Allerdings wären diese für dieses Schulprojekt nicht wirklich wichtig. Dazu zählen zum Beispiel:

- Unterstützung von Anfragen mit Binär-Inhalt
- Vernünftige Unterstützung von Charsets
- Standard-Header bei Http-Antworten
- Das Meiste im Zusammenhang mit Websockets
- Respektierung des "Accept"-Headers, siehe RFC 2616 Abschnitt 14.1
- Unterstützung von "Content Codings" (u.a. Kompression), siehe RFC 2616 Abschnitt 14.41
- Unterstützung von "Transfer Codings", siehe RFC 2616 Abschnitt 3.6
- Persistente Verbindungen zu Browsern, siehe RFC 2616 Abschnitt 8.1
- Die Http "OPTIONS"-Methode, siehe RFC 2616 Abschnitt 9.2

Vorgehensweise

Zuerst habe ich mir dieses Projekt ausgesucht - da das Http-Protokoll ein wichtiger Baustein
des heutigen Internets ist und HTTP/1.1 ein einfaches, rein textbasiertes Protokoll ist, habe ich mich bei diesem Schulprojekt für eine einfache Implementation des HTTP/1.1-Protokolls entschieden.

Bei der Umsetzung dieses Programmierprojekts habe ich zuerst damit angefangen, die Projektstruktur einzurichten. Mit dem Build-Tool Gradle kann man einfach
Projekte aufsetzen, compilen und testen.

Nach dem Aufsetzen des Projektes habe ich angefangen, ein Logging-System zu programmieren. Danach habe ich einen "HttpReader"-Helfer umgesetzt, da das Http-Protokoll
einige immer wieder auftretende Elemente hat.

Weiter habe ich generelle Strukturen des Http-Protokolls umgesetzt wie Anfragen, Header, Status, Version und Antworten. Schließlich habe ich den Http-TCP-Socket-Server umgesetzt und die Logik für einen einfachen Datei-Server geschrieben, welcher einfach nur Dateien an den Browser versendet (mit zugehörigem MIME-Type).

Am Ende habe ich einige Stellen noch ausgebessert und den Quellcode mit Kommentaren versehen, damit auch andere den Quellcode verstehen können.

Probleme

Wirklich große Probleme gab es bei diesem Projekt nicht. Allerdings musste ich zwischendurch den "HttpReader"-Helfer stream-basiert umschreiben, da ich vorher davon ausgegangen war, dass der Browser den vollständigen Inhalt absendet und die Verbindung danach direkt schließt. Der Browser lässt die Verbindung allerdings offen, weswegen das Programm nicht warten kann, bis alle Daten aus der Verbindung gelesen wurden und stattdessen stream-basiert die Anfrage direkt auslesen muss.

Quellen

- RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1
  - Zugriff am 12.03.2025 um 10:09 Uhr CET
  - Siehe auch Datei "sources" → "rfc2616.txt"
- Common MIME types - HTTP | MDN
  - Zugriff am 12.03.2025 um 10:17 Uhr CET
  - Siehe auch Datei "sources" → "MDN_HTTP_Common_MIME_types.png"
- HTTP response status codes - HTTP | MDN
  - Zugriff am 12.03.2025 um 10:14 Uhr CET
  - Siehe auch Datei "sources" → "MDN_HTTP_HTTP_response_status_codes.png"

Falls relevant, werden im Quellcode teilweise auch direkt spezielle Quellen erwähnt.