**Fachhochschule Vorarlberg**

| | |
|---|---|
| | Datum 02.11.2022 |
| Informatik<br>Web Application Engineering<br><br>Thomas Feilhauer | Übungsblatt 4 |
| | Zu lösen bis 09.11.2022, 8:00 Uhr |

## *Vorbereitungsaufgabe (in der Übungsstunde zu lösen):*

Machen Sie sich vertraut mit dem Erstellen von HTML-Seiten:
Tutorials:
http://de.html.net/tutorials/html/
https://www.webmaster-crashkurs.de/
https://www.html-seminar.de/
https://www.w3schools.com/html/default.asp
http://www.csszengarden.com/

Zum Nachschlagen:
https://wiki.selfhtml.org/wiki/Startseite

## *Aufgabe 5: HTTP* *(3 Punkte)*

Simulieren Sie mit Hilfe von **telnet** eine **HTTP1.1**-Verbindung zu dem von Ihnen lokal installierten Apache HTTP-Server (in seiner Default-Konfiguration). Die Verbindung soll zu der folgenden (z.B. im Browser eingegebenen) **URL** (ggf. angepasst bei abweichendem Port) hergestellt werden:

**`http://localhost/index.html`**

Über die bestehende Verbindung sollen sodann (über **telnet**) HTTP-Requests an den HTTP-Server gesendet werden. Geben Sie für jede der unten angegebenen Teilaufgaben jeweils einen vollständigen HTTP-Request an, um den HTTP-Response der zur jeweiligen URL (aus dem HTTP-Request) gehörenden Ressource oder wenigstens den Response-Header auszulesen. Modifizieren Sie dazu für die einzelnen Teilaufgaben die **Anfrage-URL** nach der HTTP-Methode in der Request-Line entsprechend der Vorgabe in der jeweiligen Teilaufgabe (der Verbindungsaufbau über telnet kann für alle Teilaufgaben identisch erfolgen):

a) Bei unveränderter Eingabe der oben angegebenen URL
b) Wenn Sie **`index.html`** aus der URL weglassen
c) Wenn Sie **`http://`** aus der URL weglassen
d) Wenn Sie in der URL **`index.html`** durch **`index.asp`** ersetzen
e) Wenn Sie als URL nur **`/index.html`** angeben
f) Wenn Sie als URL nur **`index.html`** angeben

Überprüfen Sie jeweils den **Statuscode** der Antwort. Protokollieren Sie für jede der Teilaufgaben jeweils Ihre Eingaben sowie die zugehörigen Ergebnisse (Status-Line) und begründen Sie kurz das Ergebnis.

Hinweise: Eine entspr. Simulation mit curl oder anderen Tools für das Testen von TCP/IP-Requests ist typischerweise nicht sinnvoll, da diese Werkzeuge im Allgemeinen Ergänzungen/Korrekturen an der Eingabe vornehmen, um den User vor Fehlern zu bewahren. Beachten Sie darüber hinaus die Hinweise für Nutzer:Innen von MS Windows zu Aufgabe 3; falls Sie telnet auf einer virtuellen Maschine oder dem Windows-Subsystem für Linux starten und Ihr Apache HTTP-Server auf dem (Windows-)Host installiert ist, dann muss statt localhost die IP-Adresse des Hosts verwendet werden, denn der localhost in der VM adressiert die VM selbst, die eine eigene IP hat. Die IP-Adresse des Hosts können Sie z.B. über

den Konsolenbefehl ipconfig (auf dem Host) ermitteln. localhost kann nur dann verwendet werden, wenn der Apache HTTP-Server auch in der VM gestartet wurde.

## *Aufgabe 6: WebServer (6 Punkte)*

Erstellen Sie in Java einen einfachen WebServer für statische Ressourcen (HTML-Datei), der einen minimalen Umfang des HTTP-Protokolls implementiert (GET auf HTTP 1.1 genügt). Analysieren Sie hierzu zunächst einen typischen HTTP-Request, indem Sie z.B. mit telnet einen Request an einen Server schicken und dessen Antwort auswerten (vgl. Aufgabe 5). Ihr WebServer soll in der Lage sein, einfache HTML-Dokumente aus einem Dateisystem per HTTP bereitzustellen.
Funktionsweise des einfachen WebServers:

- Belege den TCP-Port **8080** und warte auf dort eintreffende HTTP-Requests.
- Akzeptiere die eintreffenden Requests.
- Analysiere den HTTP-Request und lokalisiere das zu transferierende Dokument.
- Generiere einen entsprechenden HTTP-Response mit dem Dokument-Inhalt in der Entity und einfachem Header.
- Sende den generierten Response an den anfragenden User Agent.

Für die Implementierung des Browsers dürfen **keine High-Level Funktionen** von Java eingesetzt werden, die WebServer-Funktionalität bereitstellen (z.B. alle Klassen in **`com.sun.net.httpserver.*`** sind tabu!). Die Kommunikation soll über explizit in Java programmierte TCP-Socket-Verbindungen erfolgen und die Generierung der HTTP-Response ist im Programm zu codieren. Wenn die angeforderte Ressource fehlerfrei bedient werden kann, dann sollte der Statuscode 200 an den Client zurückgegeben werden; falls die angeforderte Ressource auf dem Server nicht vorhanden ist, dann sollte der Statuscode 404 zurückgegeben werden und bei allen anderen auftretenden Fehlern der Statuscode 500.

Überprüfen Sie Ihr Programm, indem Sie über einen Standard-Web-Browser ein einfaches HTML-Dokument von Ihrem WebServer anfordern und es anschließend im Browser anzeigen lassen. Geben Sie das **Web-Root-Verzeichnis** an, in dem die Web-Ressourcen abgelegt werden können und wählen Sie dieses so, dass es relativ zu der main-Class Ihres Programms durch Ihre Anwendung eingelesen wird, z.B. könnte es in dem (beliebigen) Verzeichnis, in dem Ihre main-Class File liegt, einen Ordner geben, der „webroot“ heißt! Also kein absolutes Verzeichnis angeben, das z.B. mit C:\ beginnt!