Übungsblatt 02 – Application Layer

Aufgabe A – HTTP(S) & HTML

1. Navigieren sie mithilfe der Kommandozeile in den `shell_tutorial` Ordner. Kopieren sie sie die Datei in `shell_tutorial.md` und bennen sie die Kopie in *index.html* um.

2. Wir gestalten im Folgenden eine kleine Website mit den Inhalten ihres Cheat-Sheets:

   a) Öffnen sie die `index.html` mithilfe von *vim*.

   b) Da die Website in *HTML* gehalten ist, sollten wir dies festhalten. Mit dem Tag `<!DOCTYPE html>` in der ersten Zeile kann dies erfolgen.

   c) Jedes weitere Tag muss einen Beginn und Ende haben. D.h. `<TAG>` setzt den Start und `</TAG>` setzt das Ende. Legen sie ein `html`-Tag an. Also `<html>` in die zweite Zeile und `</html>` in die letzte Zeile.

   d) Im Header werden Metainformationen festgehalten. Setzen sie ein Head-Tag. In das Head-Tag setzen sie ein Title-Tag.

   e) Der Body definiert den eigentliche Inhalt der Website, hier liegen die sichtbaren Inhalte. Legen sie einen Body um ihre genutzen Kommandos an.

   f) `<h1>` gibt Überschriften an, wobei `<h1>` die größtmögliche Überschrift ist.

   g) Mit `<p>` können Paragraphen angelegt werden.

   h) Mit `<br>` erzwingen sie eine Linebreak (Zeilenumbruch).

   i) Gestalten sie ihr Cheat-Sheet zu einer kleinen simplen Website.

   j) Überprüfen sie, ob ihr HTML alle notwendigen Tags des Listing aufweist:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>Page Title</title>
5  </head>
6  <body>
7
8  <h1>Shell Cheat Sheet</h1>
9  ...
10 <h3>Basics</h1>
11 <p>ls</p>
12 ls -la <br>
13 ls -lisa <br>
14 ...
15 <h3>vi/vim</h1>
16 </body>
```

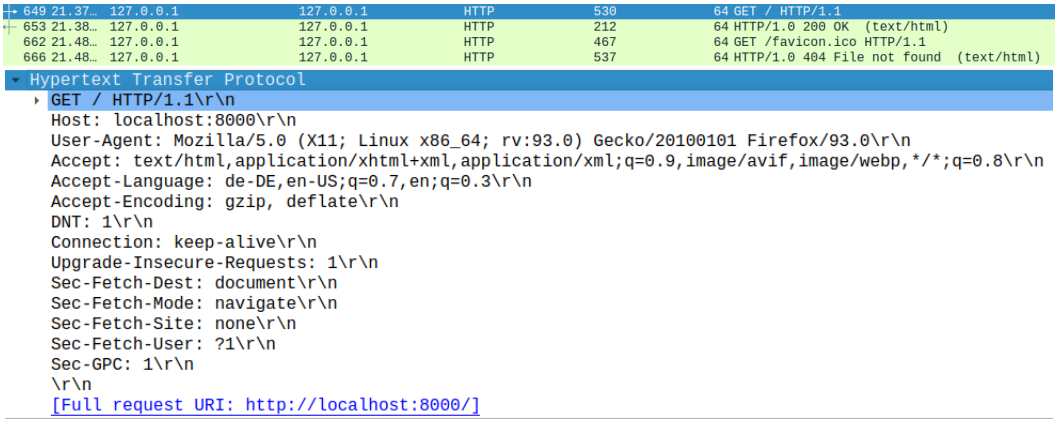
```
17 </html>
```

3. Python liefert ihnen eine rudimentären Webserver frei Haus. Starten sie im Ordner der `index.html` Datei folgendes Kommando:

```
1 python3.8 -m http.server 8000
2 #Fuer die alte VM:
3 python3.7 -m http.server 8000
```

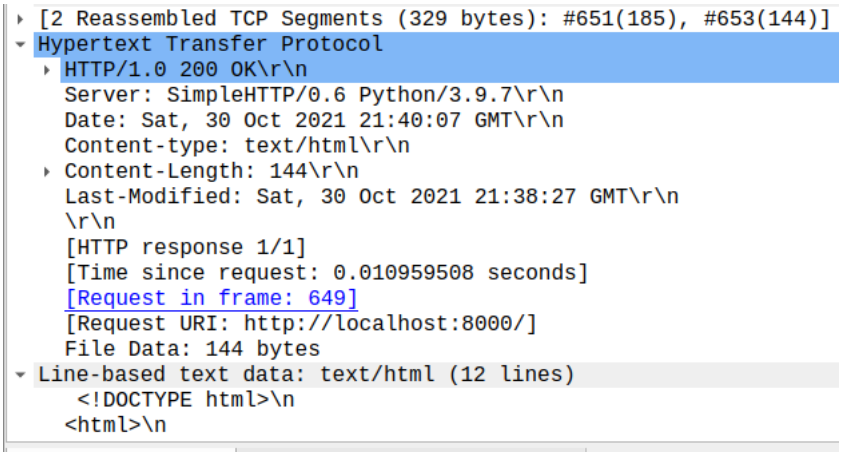
4. Rufen sie die Webseite mit `http://localhost:8000` im Browser auf.

5. Folgendes kann beobachtet werden:



   a) Der Webserver arbeitet mit *HTTP*. Zum abrufen einer HTTP-Anfrage wird die *GET*-Methode genutzt. Lesen sie genau, wie der *GET*-Request aussieht. Für uns sind die ersten zwei Zeilen.

   b) Auf den Request folgt die Antwort des Servers via *HTTP*-Response.



Lesen sie den Response genau.

6. Verbinden sie sich nun via Kommandozeilen-Programm *telnet* mit unserem lokalen *HTTP*-Server.

```
telnet localhost 8000
```

   Mit *telnet* können sie sich auf verschiedenste Server verbinden. Es dient hier lediglich, um eine Verbindung zum Webserver herzustellen.

7. Nachdem die Verbindung hergestellt wurde wollen wir ein *HTTP*-Request absetzen. Tippen sie die ersten zwei Zeilen des *GET*-Requests ab – lassen sie jedoch die Zeichen `\r\n` weg.

8. Wenn sie zu einem Server eine Verbindung aufbauen, wird serverseitig ein Timeout gestartet, so das, wenn nicht innerhalb einer gewissen Zeitspanne eine Anfrage kommt, der Server die Verbindung beendet. Wenn sie etwas umfangreichere Befehle an den Server senden müssen, oder das Ganze ohne manuellem Eintippen automatisieren wollen, können Sie das Tool *netcat* nutzen.

9. Mit dem Werkzeug *netcat* können Website direkt über die Kommandozeile aufgerufen werden.

```
nc -v localhost 8000
```

   a) Schreiben sie die gleichen HTTP-Befehle zum Abruf der Webseite jetzt in eine lokale Textdatei Diesmal.
   Dise können sie mithilfe der *printf* Funktion beispielsweise lösen:

```
printf "GET \ HTTP1.1 ..." > request
printf "GET / HTTP/1.1\nHOST: localhost:8000\n\n"
cat request | nc localhost 8000
```

   Alternativ könne sie dies auch via *vim* erledigen.

   b) Lassen sie sich den Inhalt der Datei auf der Kommandozeile nach Std-Out ausgeben (*cat*).

   c) Leiten sie diese Ausgabe mittels einer *Pipe* (`|`) als Eingabe in den Befehl *netcat* um. Rufen sie *netcat* dabei mit Parametern so auf, dass es eine Verbindung wieder zum gleichen Webserver und Port wie bisher aufbaut.
   Wenn sie alles richtig gemacht haben, sehen sie wieder die gleiche Ausgabe.

Aufgabe B – Python

1. Starten sie eine interaktive Python-Session.

```
python
```

2. Python als Taschenrechner.
   a) Berechnen sie folgende Terme:
      - $2 + 3$
      - $3.5 + 4.5$
      - $3 + 3.5$

      Lassen sie sich mit *type()* den jeweiligen Datentyp ausgeben.
   b) Deklarieren sie folgende Variablen:
      - Variable a mit Wert 2
      - Variable b mit Wert 3
      - Variable c als Summe von a und b
      - Eine Variable $s1$ die den Inhalt: *Netzwerke* und eine Variable $s2$ mit dem Inhalt *verteilte Systeme* enthält.
      - Fügen sie in der Variablen s die beiden Texte $s1, s2$ mit dem + Operator zusammen.

3. Mithilfe der *print()* Funktion können Ausgaben auf der Kommandozeile ausgegeben werden.
   a) „printen“ sie die Zeile „Hallo, Welt!“ auf der Kommandozeile.

4. Kontrollstrukturen:

```
if x % 2:
  print("x is even")
else:
  print("x is odd")
```

   a) Schreiben und testen sie eine Block der prüft, ob zwei Variablen numerisch identisch sind („==“).
   b) Schreiben und testen sie eine Block der eine Ausgabe ausführt, wenn die erste Variable größer als die zweite ist.

5. For-Schleifen:

```
for i in range(X) # X ist hier die Anzahl der Wiederholungen
```

a) Schreiben und testen sie mithilfe eine For-Schleife, die ihnen 10 mal „Hallo, Welt!“ ausgibt.

b) Schreiben und testen sie mithilfe einer For-Schleife die Gaußsche Summenformel. Also die Summe der Zahlen von 0 bis m

$$0 + 1 + 2 + ... + m = \sum_{k=0}^{m} k$$

6. While-Schleifen:

```
while i <= n: # X ist hier die Anzahl der Wiederholungen
```

a) Schreiben sie eine *While*-Schleife, die alle Zahlen kleiner der Variablen c nacheinander ausgibt.

7. Funktionen

```
def foo(): # Funktion ohne Argument
def bar("Blub"): #Funktion mit Argument hard coded
def bar2(m): Funktion mit Argument als Variable
def square(x): # Funktion mit Argument und Rückgabewert
  return x**2
```

a) Schreiben sie eine Funktion `euclid` die den euklidischen Algorithmus implementiert. Nutzen sie eine Funktiosdeklaration.