10 Ein intelligenter Server für Webseiten

Unser nächster Webserver soll uns nun persönlich begrüßen. Im Browserfenster soll nicht mehr einfach das unpersönliche “Hallo Welt” stehen wie in Kapitel 8, sondern ein “Hallo Irmin” oder “Hallo Bernd”, je nachdem wie unser Name lautet. Um das leisten zu können, muss der Server natürlich in Erfahrung bringen, von wem er gerade eine Anfrage erhält. Wie können wir das erreichen? Nun, in Kapitel 7 hatten wir schon gesehen, wie der Client über den GET-Parameter des HTTP-Requests Informationen an den Server senden kann; damals hatten wir damit den Namen der gewünschten Datei übermittelt. Genauso wollen wir jetzt vorgehen; allerdings senden wir nicht den Namen einer Datei, sondern unseren eigenen Namen.

In die Adresszeile des Browsers schreiben wir also einfach hinter die IP-Adresse des Servers ein /-Zeichen und dahinter unseren Namen. Die Eingabe sieht dann z. B. so wie in Abb. 1. aus:

Abb. 1

Der Browser teilt diese Eingabe auf in die Host-Adresse, hier also die IP-Adresse sowie den GET-Parameter `'/Kai-Uwe'`. Diese Informationen sendet er nun bei seinem Request an den Server.

Was muss unser Server-Programm jetzt damit tun? Zunächst muss es den Namen aus dem Request-String herausschneiden. Das geht z. B. so:

```
getparam = request.split(' ')[1] # Element 1 von ['GET', '/name', ... ]
myname = getparam[1:] # '/' gehört nicht zum Namen
```

In der ersten Zeile wird der Request-String zunächst mit Hilfe der **`split`**-Methode in eine Liste zerlegt. Deren Elemente sind genau die Teilketten des Request-Strings, die durch ein Leerzeichen getrennt sind. Das Element mit dem Index 1 ist dann gerade unser GET-Parameter. Wenn wir noch das erste Zeichen (nämlich `'/'`) von `getparam` mit `[1:]` weglassen, haben wir unseren eigenen Namen isoliert.

Der zu sendende HTML-String lautet jetzt:

```
'<html><body><h1>Hallo '+myname+'</h1></body></html>'
```

Zur besseren Übersicht fassen wir dies zusammen zu einer Funktion `html(request)`:

```
def html(request):
  getparam = request.split(' ')[1] # Element 1 von ['GET', '/name', ... ]
  myname = getparam[1:] # '/' gehört nicht zum Namen
  return '<html><body><h1>Hallo '+myname+'</h1></body></html>'
```

In vielen Fällen können wir davon ausgehen, dass der Request aus höchstens 1000 Zeichen besteht. Bei unserem Namens-Server hängt die Länge des Requests jedoch von der Eingabe des Nutzers ab. In solchen Fällen ist es nicht sicher, dass der Request aus maximal 1000 Bytes besteht. Sind es mehr Bytes, würde der Request nur unvollständig empfangen werden, und die Kommunikation zwischen Client und Server wäre gestört. Deswegen wollen wir die Zeile

```
request = connection.recv(1000)
```

ersetzen durch `request = getrequest(connection)` ; dabei wird `getrequest` so definiert:

```
def getrequest(connection):
  result = ''
  block_size = 64 # nur zum Test so klein, sonst z. B. 512
  while True:
    data = connection.recv(block_size)
    result += str(data, 'UTF-8') # Blöcke verketten
    if result.find('\r\n\r\n') >= 0: # Ende des Requests gefunden
      break
  return result
```

Innerhalb einer Schleife wird hier der Request in einzelnen Blöcken einer bestimmten Maximalgröße (`block_size`) empfangen. Hier werden die einzelnen Blöcke zunächst in Strings konvertiert; diese wiederum werden sukzessive zu einem Gesamt-String `result` verbunden. Die Schleife muss abbrechen, wenn das Ende des Requests erreicht ist. Wir wissen bereits (vgl. K7.2), dass ein Request immer durch zwei CR-LF-Escape-Sequenzen abgeschlossen wird; diese sind es ja, welche für die erforderliche Leerzeile am Ende eines Request sorgen. Die String-Methode `find` sucht nun in dem String `result` nach der Zeichenkette `'\r\n\r\n'`. Wird sie fündig, ist der Rückgabewert größer oder gleich 0. In diesem Fall wird die Schleife abgebrochen.

Für den HTTP-Header benutzen wir dieselbe Funktion wie beim Temperatur-Server.

Sie können nun diese Ergänzungen bzw. Änderungen an dem Temperatur-Server-Programm vornehmen (oder einfach die Datei `sta_name_server_0.py` starten. Testen Sie das Programm mit verschiedenen Namen. Probieren Sie auch einmal aus: Wie verhält sich das Programm bei Doppelnamen ohne Bindestrich? Wie verhält es sich, wenn Sie gar keinen Namen eingeben? Versuchen Sie, Ihre Beobachtungen zu erklären!