24.13 Exkurs: Webserver Lighttpd – Installation, Konfiguration und Test

Wenn Sie Webseiten, Web-Applikationen oder ein CMS (Content Management System wie DokuWiki) lokal testen wollen, dann benötigen Sie einen Webserver wie *Lighttpd* als schnellen Webserver mit geringem Speicherverbrauch.

Lighttpd – genannt Lighty – mit hinreichend einfacher sowie überschaubarer Konfiguration ist eine Alternative zum Apache-Webserver.

Inhaltsverzeichnis

24.13.1 Hinweise

Die folgenden Hinweise und Anregungen sollten Sie beachten:

- In der vorliegenden Beschreibung für Linux Mint 20.2 wird die Installation, Konfiguration und ein grundlegender Test des Webservers Lighttpd beschrieben.
- Es wird die Version 1.4.55 des Webservers Lighttpd eingesetzt.
- Sie können für das Anlegen und die Änderung aller Konfigurationsdateien unter Root-Rechten einen beliebigen Text-Editor verwenden.
- Die Dokumentation zum Webserver sollte ebenso installiert werden, damit Sie u.a. Zugriff auf die Dokumentation der einzelnen *Module* erhalten, welche für die *Konfiguration* der Module unerlässlich ist.
- Die Pfade zur Dokumentation der einzelnen Module finden Sie in den Konfigurationsdateien der einzelnen Module.
- Die (Basis-)Funktionalität des Webservers können Sie durch einzelne Module erweitern.
- Bei einigen Befehlen in der Konsole oder bei Link-Angaben für den Webbrowser ist der Benutzername *hans* der Testumgebung des Autors stets durch Ihren Benutzernamen zu ersetzen!

24.13.2 Installation Webserver Lighttpd

Der Webserver *Lighttpd* kann über die Anwendungsverwaltung installiert werden oder Sie geben in einem Terminal folgende Zeilen ein:

```
$ sudo apt-get update && apt-get upgrade
$ sudo apt-get install lighttpd
```

Anschließend sollten Sie auch die Webserver-Dokumentation installieren:

```
$ sudo apt-get install lighttpd-doc
```

Die installierte Version von Lighttpd zeigen Sie so an:

```
$ lighttpd -v
lighttpd/1.4.55 (ssl) - a light and fast webserver
```

24.13.3 Webserver-Status und Webserver-Start

Der Webserver wird nach der Installation automatisch gestartet. So können Sie seinen Status jederzeit kontrollieren:

```
$ service lighttpd status
```

oder

```
$ /etc/init.d/lighttpd status
```

Achtung! Der Webserver Lighttpd wird bei jedem Systemstart *automatisch* gestartet (Standard). Das können Sie mit diesen beiden Befehlen ändern:

```
$ sudo update-rc.d -f lighttpd remove  ' → Lighttpd aus der Autostart-Liste entfernen
  Removing any system startup links for /etc/init.d/lighttpd …
```

```
$ sudo update-rc.d lighttpd defaults   ' → Lighttpd der Autostart-Liste wieder hinzufügen
  Adding system startup for /etc/init.d/lighttpd ...
```

Option:
Wenn Sie den Webserver beim Systemstart nicht automatisch starten, können Sie den HTTP-Server jederzeit temporär als Hintergrund-Prozess starten:

Variante 1:

```
$ lighttpd -f /etc/lighttpd/lighttpd.conf
```

Hintergrund-Prozess beenden:

```
$ pidof lighttpd
$ sudo kill PID
```

Variante 2:

```
$ lighttpd -D -f /etc/lighttpd/lighttpd.conf (→ ohne Ablösung von der Konsole)
```

Den so gestarteten Hintergrund-Prozess beenden Sie mit CTRL+X oder Terminal-Fenster schließen.

Wenn Sie ständig mit dem Webserver Lighttpd arbeiten, dann lohnt das Anlegen von Startern auf dem Desktop für ausgewählte Aktionen wie zum Beispiel das Starten oder das Stoppen.

24.13.4 Webserver – Test

Nach der Installation werden Sie mit folgendem Befehl

```
$ cat /etc/passwd | cut -d: -f1 | grep www-data
www-data
```

feststellen, dass automatisch ein neuer Benutzer www-data angelegt worden ist, unter dem der Webserver Lighttpd läuft. Es existiert auch eine gleichnamige Gruppe.

Sie können den automatisch gestarteten Webserver mit der bereits vorhandenen (Basis-)Konfiguration testen, indem Sie die während der Installation automatisch erzeugte Webseite *index.lighttpd.html* – gespeichert ab Version Mint 18 im Standard-Webordner */var/www/html* – in einem Webbrowser öffnen.

Die folgenden Aufrufe der (statischen) Webseite im Webbrowser Ihrer Wahl sind gleichwertig, da standardmäßig nach einer Index-Datei im Standard-Webordner gesucht wird. :

```
http://127.0.0.1  oder  http://localhost  oder
http://127.0.0.1/index.lighttpd.html
oder
http://localhost/index.lighttpd.html oder
http://192.168.2.106 (IP des Server-Hosts → $ ifconfig.)
```

Achtung: Diese Adressen funktionieren nur, wenn Sie das Modul *userdir* noch nicht aktiviert haben!

Im Webbrowser sollte der Inhalt der Index-Datei so angezeigt werden:

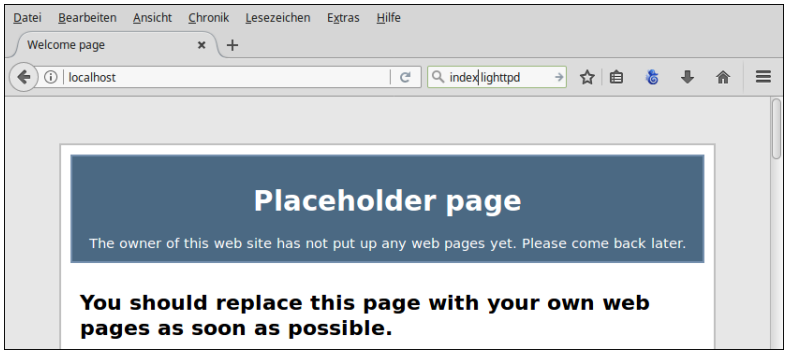

Abbildung 24.13.4.1: Ausschnitt aus dem Inhalt der Standard-Index-Datei

Lesen Sie sich den Text in der 'Platzhalter-Webseite' aufmerksam durch, da er wichtige Informationen zum Webserver enthält.

24.13.5 Dokumentation

Eine umfangreiche Dokumentation (Stand: 14.Juni 2022) zum Webserver Lighttpd finden Sie unter:

```
LINK1:          https://www.lighttpd.net/
LINK2:          https://wiki.ubuntuusers.de/lighttpd/
```

Hilfen zum Webserver Lighttpd gibt es auch hier:

```
LINK3:          https://redmine.lighttpd.net/projects/lighttpd/wiki
LINK4:          https://www.elektronik-kompendium.de/sites/raspberry-pi/1905271.htm
KONSOLE:        $ lighttpd -h
KONSOLE:        $ ls -l /usr/share/doc/lighttpd
```

24.13.6 Log-Dateien

Die letzten Einträge der *aktuellen* (Fehler-)Log-Datei des Webservers können Sie sich mit dem folgenden Befehl in einer Konsole ansehen:

```
$ sudo tail -f /var/log/lighttpd/error.log
```

Ältere Log-Dateien finden Sie ebenso im Verzeichnis /var/log/lighttpd/.

24.13.7 Webserver-Steuerung

Die folgenden Aufrufe in einem Terminal steuern den Webserver Lighttpd mit den in der Liste aufgeführten Parametern:

```
$ sudo /etc/init.d/lighttpd {start|stop|restart|reload|force-reload|status}
```

Alternative:

```
$ sudo service lighttpd {start|stop|restart|reload|force-reload|status}
```

Beispiele:

```
$ /etc/init.d/lighttpd status
$ sudo service lighttpd reload
```

24.13.8 Konfiguration des Webservers Lighttpd

Die (Basis-)Konfigurationsdatei für den Webserver Lighttpd ist /etc/lighttpd/lighttpd.conf. Ohne Änderung der Basis-Konfiguration liefert der Webserver nur statische HTML-Seiten aus dem Verzeichnis /var/www/html an den Web-Browser aus!

- An der (Basis-)Konfigurationsdatei /etc/lighttpd/lighttpd.conf werden Änderungen vorgenommen!

- Es werden die Konfigurationsdateien der vorgesehenen Module in der angegebenen Weise geändert und aktiviert.
- Mit dem erneuten Einlesen der (Basis-)Konfigurationsdatei wird sichergestellt, dass die Server-Konfiguration um die Modul-Funktionalität erweitert wird.

In der (Basis-)Konfigurationsdatei (Mint ≥ 20.1) bewirkt die folgende Include-Anweisung, dass aktivierte Modul-Konfigurationen aus dem Ordner /etc/lighttpd/conf-enabled in die (Basis-)Konfigurationsdatei eingefügt werden:

```
include "/etc/lighttpd/conf-enabled/*.conf"
```

Generell gilt: Um eine (geänderte) Konfigurationsdatei auf (Syntax-)Fehler zu überprüfen, können Sie den folgenden Befehl in einer Konsole benutzen, mit dem eine Fehlkonfiguration sehr schnell erkannt wird oder die Korrektheit der Syntax bestätigt wird, wie die folgenden zwei Beispiele zeigen:

```
$ lighttpd -t -f /etc/lighttpd/lighttpd.conf
2017-05-11 18:34:40: (configfile.c.957) source: /etc/lighttpd/lighttpd.conf line: 83 pos: 1 parser failed
somehow near here: (EOL)
```

```
$ lighttpd -t -f /etc/lighttpd/lighttpd.conf
Syntax OK
```

24.13.8.1 Änderung der Basis-Konfiguration des Webservers Lighttpd

Sichern Sie zuerst das Original der Basis-Konfigurationsdatei von Lighttpd:

```
$ sudo cp /etc/lighttpd/lighttpd.conf /etc/lighttpd/lighttpd.conf.old
```

Ändern Sie dann den Inhalt der Basis-Konfigurationsdatei lighttpd.conf auf den u.a. Inhalt mit Kommentaren. Speichern Sie danach die geänderte Datei:

```
$ sudo geany /etc/lighttpd/lighttpd.conf ' Editoren: nano, gedit, bluefish, bracket ...
```

```
# Determination of the modules that are to be activated immediately

server.modules = (
                "mod_indexfile",
                "mod_access",
                "mod_alias",
                "mod_redirect",
)

# Server-Configuration:

server.document-root        = "/var/www/html"
server.upload-dirs          = ( "/var/cache/lighttpd/uploads" )
server.errorlog             = "/var/log/lighttpd/error.log"
server.pid-file             = "/run/lighttpd.pid"
server.username             = "www-data"
server.groupname            = "www-data"
server.port                 = 80

# strict parsing and normalization of URL for consistency and security
# https://redmine.lighttpd.net/projects/lighttpd/wiki/Server_http-parseoptsDetails
# (might need to explicitly set "url-path-2f-decode" = "disable"
#  if a specific application is encoding URLs inside url-path)

server.http-parseopts = (
  "header-strict"           => "enable",  # default
  "host-strict"             => "enable",  # default
  "host-normalize"          => "enable",  # default
  "url-normalize-unreserved"=> "enable",  # recommended highly
  "url-normalize-required"  => "enable",  # recommended
  "url-ctrls-reject"        => "enable",  # recommended
  "url-path-2f-decode"      => "enable",  # recommended highly (unless breaks app)
 #"url-path-2f-reject"      => "enable",
  "url-path-dotseg-remove"  => "enable",  # recommended highly (unless breaks app)
 #"url-path-dotseg-reject"  => "enable",
 #"url-query-20-plus"       => "enable",  # consistency in query string
)

# Default extensions for index files

index-file.names            = ( "index.php", "index.html", "index.gambas" )
```

```
# Default forbidden file extensions

url.access-deny             = ( "~", ".inc" )

# Do *not* consider files with the following extensions as static content

static-file.exclude-extensions = ( ".php", ".pl", ".py", ".cgi", ".gbs", ".gbw", ".gambas")
compress.cache-dir          = "/var/cache/lighttpd/compress/"
compress.filetype           = ( "application/javascript", "text/css", "text/html", "text/plain" )

# Default listening port for IPv6 falls back to the IPv4 port
# Use ipv6 if available
# include_shell "/usr/share/lighttpd/use-ipv6.pl " + server.port
# Set allowed MIME types

include_shell "/usr/share/lighttpd/create-mime.conf.pl"

# Enabled module configurations from the /etc/lighttpd/conf-enabled folder will be
# inserted into the (main) configuration file.

include "/etc/lighttpd/conf-enabled/*.conf"

#server.compat-module-load   = "disable"

server.modules += (
               "mod_compress",
               "mod_dirlisting",
               "mod_staticfile",
)
```

Nach jeder Änderung einer Konfigurationsdatei müssen Sie die Konfigurationsdatei stets neu einlesen, damit die Änderungen wirksam werden. Mit dem Parameter *force-reload* wird die aktuelle Konfiguration neu eingelesen und der Webserver neu gestartet.

```
$ sudo service lighttpd force-reload        # Ohne Quittung, wenn der Reload erfolgreich war
$ sudo /etc/init.d/lighttpd force-reload    # Gute Alternative, da mit Ausgabe eines Kommentars
Reloading lighttpd configuration (via systemctl): lighttpd.service.
```

24.13.8.2 Erweiterung der Funktionalität des Webservers

Die Funktionalität des Webservers können Sie durch Module erweitern, damit bestimmte Dienste zuverlässig zur Verfügung stehen. Die Konfiguration des Webservers Lighttpd wird im Kapitel dabei so vorgenommen, dass folgende Server-Funktionalitäten erzielt werden:

- Modul userdir: Ein System-Benutzer verwaltet alle eigenen Webseiten und Skripte im seinem Home-Verzeichnis in einem speziellen (Standard-)Webordner /$HOME/public_html.
- Modul cgi: CGI-Skripte der Programmiersprachen Gambas (Skripte und Webpages), Perl und Python werden in einem speziellen (Unter-)Ordner ~/public_html/cgi-bin ausgeführt
- Modul fastcgi: PHP-Skripte werden in ~/public_html ausgeführt
- Modul ssi: SSI-Skripte werden in ~/public_html ausgeführt

Sie finden auf der Website *https://redmine.lighttpd.net/projects/1/wiki/Docs* u.a. eine Übersicht der Module als Link-Liste, so dass Sie sich schnell über die Modul-Beschreibung und die Konfigurationsoptionen zum ausgewählten Modul informieren können!

Es wird in den nächsten Abschnitten detailliert beschrieben, wie Sie die Basis-Konfiguration durch die Aktivierung einzelner Module Schritt für Schritt erweitern können, um eine bestimmte Funktionalität des Webservers zu erzielen. Viele hinreichend vorkonfigurierte Konfigurationsdateien finden Sie im rot gefärbten Ordner /etc/lighttpd/conf-available:

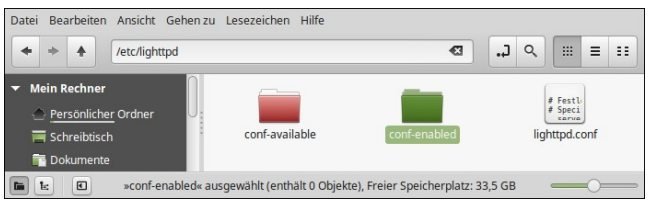

Abbildung 24.13.8.2.1: Lighttpd – Ort der Konfigurationsdateien

Hinweis: Sie sollten nur die Module aktivieren, die Sie auch tatsächlich benötigen!

Um den Funktionsumfang des Webservers durch den Einsatz eines *bestimmten* Moduls zu erweitern, hat es sich bewährt, die folgende Schrittfolge für das Aktivieren eines Moduls abzuarbeiten:

(1) Modul konfigurieren durch das Bearbeiten der Modul-Konfigurationsdatei mit Root-Rechten im Ordner etc/lighttpd/conf-available.
(2) Modul aktivieren.
(3) Basis-Konfigurationsdatei neu einlesen.
(4) Funktionalität des Moduls im Webbrowser Ihrer Wahl testen.

In den nächsten Absätzen wird die Konfiguration ausgewählter Module des Webservers *Lighttpd* ausführlich beschrieben.

24.13.8.3 Konfiguration Modul userdir

Auf das Verzeichnis */var/www/html* – in dem der Webserver Lighttpd statische oder dynamische Webseiten erwartet – können Sie nur mit Root-Rechten zugreifen, um dort Dateien anzulegen, zu ändern oder zu löschen!

Das Modul *userdir* bietet eine einfache Möglichkeit, ein spezielles Verzeichnis in Ihrem Home-Verzeichnis mit dem standardisierten Namen *public_html* in den globalen Namespace des Webservers einzubinden (→ symbolische Verknüpfung).

Legen Sie als System-Benutzer zuerst den Webordner in Ihrem Home-Verzeichnis an:

```
$ mkdir /home/$USER/public_html
```

Das Recht für den Webordner public_html wurde automatisch auf 755 oder drwxr-xr-x als Rechte-String gesetzt. Zukünftig können Sie Ihre eigenen Webseiten (HTML) und Web-Skripte (Gambas, Perl, Python, PHP, SSI) in diesen Webordner (oft als DOCUMENT_ROOT bezeichnet) kopieren oder auch dort als Skript schreiben.

Im folgenden Abschnitt wird die o.a. Schrittfolge für den Einsatz eines Moduls am Beispiel vom Modul *userdir* exemplarisch umgesetzt:

(1) Modul konfigurieren durch das Bearbeiten der passenden Konfigurationsdatei

Lesen Sie sich stets zuerst die Dokumentation zum Modul durch:

```
$ xed /usr/share/doc/lighttpd/userdir.txt
```

Sichern Sie dann das Original der Modul-Konfigurationsdatei:

```
$ sudo cp /etc/lighttpd/conf-available/10-userdir.conf /etc/lighttpd/conf-available/10-userdir.conf.old
```

Das Modul *userdir* wird durch das Bearbeiten Datei /etc/lighttpd/conf-available/10-userdir.conf mit dem Editor Ihrer Wahl konfiguriert.

```
$ sudo xed /etc/lighttpd/conf-available/10-userdir.conf
```

Ändern Sie den Inhalt der Konfigurationsdatei 10-userdir.conf auf den u.a. Inhalt.

```
## The userdir module provides a simple way to link user-based directories into the global namespace of the
webserver.
##
## --- MODULE: USERDIR ---
##
## Documentation: /usr/share/doc/lighttpd/userdir.txt
##
## Aktivierung des Moduls 'userdir'
## Activation of the module 'userdir'

server.modules += ( "mod_userdir" )

## Pfadangabe zum Webordner im Home-Verzeichnis
## Path to the web folder in the home directory

userdir.path = "public_html"
```

```
## Bestimmten Benutzern den Zugriff auf den Webordner public_html verbieten
## Certain users prohibit access to the public_html web folder

userdir.exclude-user = ( "root", "postmaster" )

##
## --- END OF MODULE: USERDIR ---
```

Speichern Sie dann die geänderte Modul-Konfigurationsdatei.

(2) Modul aktivieren

Das Aktivieren des Moduls erfolgt durch den Befehl:

```
$ sudo lighttpd-enable-mod userdir
...
Run /etc/init.d/lighttpd force-reload to enable changes
```

Wenn Sie eine Modul-Konfigurationsdatei aktivieren, dann wird im Ordner /etc/lighttpd/conf-enabled eine Verknüpfung zur Modul-Konfigurationsdatei als symbolischer Link gespeichert.

Hinweis: Mit dem folgenden Befehl können Sie ein bestimmtes Modul de-aktivieren, sofern das erforderlich ist:

```
$ sudo lighttpd-disable-mod modulname
```

(3) Konfigurationsdateien neu einlesen

Wie bereits erwähnt, müssen Sie nach jeder Änderung der Konfiguration die geänderte (Basis-)Konfigurationsdatei neu einlesen, wie es nach dem Aktivieren auch mit '*Run /etc/init.d/lighttpd force-reload to enable changes*' gefordert wurde:

```
$ sudo /etc/init.d/lighttpd force-reload
```

(4) Funktionalität des aktivierten Moduls userdir im Webbrowser Ihrer Wahl testen

Übung: Legen Sie in Ihrem Webordner ~/*public_html* die (Text-)Datei *test.html* an:

```
$ geany /home/$USER/public_html/test.html ' Leere HTML-Datei erzeugen
```

Tragen Sie folgenden Inhalt ein:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>TEST.HTML</title>
    <meta charset="utf-8">
    <style>
      body {background-color: #E2E2E2; font-family: Arial; font-size:10px; color:#000000;}
      h1 {text-align: center; font-family: Arial; font-size: 32px; color: blue;}
    </style>
  </head>
  <body>
    <h1>Test-HTML5-Datei <br />im Webordner ~/public_html</h1>
  </body>
</html>
```

Hinweise

- Der Benutzer www-data, unter dessen Rechten der Webserver lighttpd läuft, benötigt mindestens Lese-Rechte auf Ihren Webordner ~/public_html und die darin enthaltenen Dateien.
- Prüfen Sie, ob für alle Verzeichnisse und Dateien die erforderlichen Datei-Rechte gesetzt sind.

Die neue Datei test.html besitzt die Datei-Rechte 644, was dem Standard für (statische) HTML-Dateien im Webordner entspricht. Das können Sie stets so schnell prüfen:

```
$ cd $HOME/public_html && ls -l | grep test.html && stat -c "%a %n" $HOME/public_html/test.html
```

Die Ausgaben zeigen Ihnen nicht nur den Besitzer an, sondern auch die Rechte als Rechte-String und in oktaler Notation:

```
-rw-rw-r--  1 hans     hans        431 Aug 12 11:03 test.html
664 /home/hans/public_html/test.html
```

Abschließend können Sie die (statische) HTML-Datei im Webbrowser Ihrer Wahl mit dem folgenden URL in einer speziellen Syntax aufrufen, wobei USERNAME durch den eigenen Benutzernamen ersetzt werden muss:

```
http://localhost/~USERNAME/test.html
http://127.0.0.1/~USERNAME/test.html
```

Abbildung 24.13.8.3.1: Anzeige der Test-Datei im Webrowser für den Benutzer 'hans'

Hinweis: Alle erzeugten Webseiten sollten den *HTML5-Standard* erfüllen, für den die Grundstruktur stets gleich ist:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>Webserver Lighttpd</title>
    <meta charset="utf-8">
    <!-- Einsatz von Inline-CSS – im Gegensatz zu einer separaten CSS-Datei -->
    <style>
      body {background-color: #C3DDFF; font-family: Arial; font-size:14px; color:#000000;}
      h1 {text-align: center; font-family: Verdana; font-size: 32px; color: #FF0000;}
    </style>
  </head>
  <body>
    <br />
    <h1>HTML5-Datei (HTML5-Standard)</h1>
  </body>
</html>
```

Der HTML-Validator auf → https://www.freeformatter.com/html-validator.html zeigt für die o.a. HTML-Datei:

```
The document is valid and conforms to best practices and standards!
```

Alternativ können Sie auch diesen Validator einsetzen → https://validator.w3.org/#validate_by_input.

24.13.8.4 Konfiguration Modul cgi

Für den Fall, dass Sie auch dynamische und interaktive Webseiten als CGI-Skripte – zu denen auch Gambas-Webpages *.gambas oder Perl-Skripte oder Python-Skripte gehören – einsetzen wollen, müssen Sie das Modul *cgi* konfigurieren und aktivieren.

(1) Modul konfigurieren durch das Bearbeiten der passenden Konfigurationsdatei
Lesen Sie sich die Dokumentation zum Modul durch:

```
$ sudo xed /usr/share/doc/lighttpd/cgi.txt
```

Sichern Sie das Original der Konfigurationsdatei:

```
$ sudo cp /etc/lighttpd/conf-available/10-cgi.conf /etc/lighttpd/conf-available/10-cgi.conf.old
```

Das Modul cgi wird durch das Bearbeiten Datei /etc/lighttpd/conf-available/10-cgi.conf konfiguriert:

```
$ sudo xed /etc/lighttpd/conf-available/10-cgi.conf
```

Ändern Sie den Inhalt der Konfigurationsdatei 10-cgi.conf auf diesen Inhalt:

```
## --- MODULE: CGI ---
##
## Documentation: /usr/share/doc/lighttpd/cgi.txt
##
## Aktivierung des Moduls 'cgi'
## Activation of the module 'cgi'

server.modules += ( "mod_cgi" )

## Alle ausführbaren Dateien (+x) sind CGI-Skripte ...
## All executable files (+x) are CGI scripts ...

cgi.execute-x-only = "enable"
cgi.assign = ( "" => "" )

## CGI-Skripte nur im benutzer-definierten Ordner ~/public_html/cgi-bin ausführen
## CGI scripts only run in the user-defined folder ~/public_html/cgi-bin

$HTTP["url"] =~ "^/cgi-bin/" { cgi.assign = ( "" => "" ) alias.url += ( "/cgi-bin/" => "/usr/lib/cgi-bin/" ) }

## --- END OF MODULE: CGI ---
```

(2) Modul aktivieren

```
$ sudo lighttpd-enable-mod cgi
[sudo] password for hans:
Enabling cgi: ok
Run /etc/init.d/lighttpd force-reload to enable changes
```

(3) Konfigurationsdateien neu einlesen

```
$ sudo service lighttpd force-reload
```

(4) Funktionalität des aktivierten Moduls im Webbrowser testen

Test 1: CGI-Skript in der Programmiersprache Perl – abgespeichert direkt im Webordner ~/public_html/cgi-bin. Dieser Ordner ../cgi-bin ist noch zusätzlich anzulegen. Das ist der Quelltext für das Perl-Skript ~/public_html/cgi-bin/pl.pl:

```
#!/usr/bin/perl
print "Content-type:text/html\n\n";
print "<!DOCTYPE html>";
print "<html lang=\"de\">";
print   "<head>";
print     "<title>Hallo Welt ...</title>";
print     "<meta charset=\"utf-8\">";
print     "<style>";
print       "body {background-color:#C3DDFF;font-family:Arial,Verdana,Courier;}";
print       "h2 {color:blue;}";
print     "</style>";
print   "</head>";
print   "<body>";
print     "<h2>Hallo Welt!<br />Ist das nicht ein schönes Perl-CGI-Programm?</h2>";
print   "</body>";
print "</html>";
```

Nicht vergessen, für das Perl-Skript die angegebenen Rechte zu setzen:

```
$ sudo chmod 645 $HOME/public_html/cgi-bin/pl.pl
```

Rufen Sie das Perl-Skript im Webbrowser Ihrer Wahl auf:

```
http://localhost/~USERNAME/cgi-bin/pl.pl
```

Abbildung 24.13.8.4.1: Perl-Skript

Test 2: CGI-Skript in der Programmiersprache Python – abgespeichert direkt im Webordner ~/public_html/cgi-bin. Das ist der Quelltext für das Perl-Skript ~/public_html/cgi-bin/py.py:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Import modules for CGI handling
import cgi, cgitb

print ("Content-type:text/html\n\n")
print ("<!DOCTYPE html>")
print ("<html lang=\"de\">")
print   ("<head>")
print     ("<title>Hallo Welt ...</title>")
print     ("<meta charset=\"utf-8\">")
print     ("<style>")
print       ("body {font-family:Arial,Verdana;background-color:#C3DDFF;}")
print       ("h2 {color:blue;}")
print     ("</style>")
print   ("</head>")
print   ("<body>")
print     ("<h2>Hallo Welt!<br />Ist das nicht ein schönes Python-CGI-Programm?</h2>")
print   ("</body>")
print ("</html>")
```

Setzen die angegebenen Rechte:

```
$ sudo chmod 645 $HOME/public_html/cgi-bin/py.py
```

Rufen Sie dann das Python-Skript im Webbrowser Ihrer Wahl auf:

```
http://localhost/~USERNAME/cgi-bin/py.py
```

Abbildung 24.13.8.4.2: Python-Skript

Test 3: Hinweis: Wenn Sie auch Web-Skripte *.gbw3 verwenden wollen, dann müssen Sie zum Beispiel bei Mint 20.1 ein zusätzliches Paket installieren:

```
$ sudo apt-get install gambas3-scripter
```

Getestet wird mit einem Web-Skript in der Programmiersprache Gambas – abgespeichert im Webordner ~/public_html/cgi-bin. Das ist der Quelltext für das CGI-Skript ~/public_html/cgi-bin/env.gbw3:

```
#!/usr/bin/env gbw3
<% DIM sElement AS String %>
<html>
  <h2>Umgebungsvariablen</h2>
  <table border="1" cellspacing="0" cellpadding="2">
    <tr>
      <th align="left">Name</th>
      <th align="left">Value</th>
    </tr>
    <% FOR EACH sElement IN Application.Env %>
    <tr valign="top">
      <td><%= sElement %></td><td><%= Application.Env[sElement] %></td>
    </tr>
    <% NEXT %>
  </table>
</html>

# sudo chmod 760 $HOME/public_html/cgi-bin/env.gbw3
```

Nicht vergessen: Die folgenden Rechte setzen:

```
$ sudo chmod 665 $HOME/public_html/cgi-bin/env.gbw3
```

Rufen Sie das Gambas-Web-Skript im Webbrowser Ihrer Wahl auf:

```
http://localhost/~USERNAME/cgi-bin/env.gbw3
```

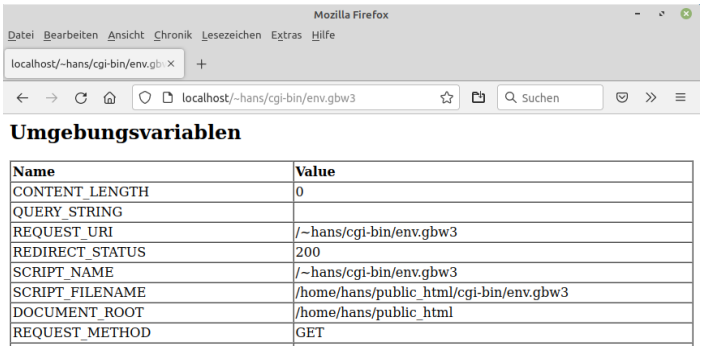

Abbildung 24.13.8.4.3: Ausgabe der Werte der Umgebungsvariablen des Webservers

Test 4: Für den vierten Test wird eine Webseite auf der Basis der Klasse Webpage (gb.web) in der IDE geladen. Sie finden das Projekt-Archiv im Downloadbereich. Die Webseite zeigt statischen Text und die Ausgabe einer JavaScript-Funktion für einen Timer.

Im Quelltext wird im Head-Bereich mit Inline-CSS (<style>…</style>) das Design der Webseite festgelegt und ebenfalls inline die JavaScript-Quelltext (<script>...</script>) notiert. Ergänzend wird gezeigt, wie Sie den Wert einer Funktion in der Programmiersprache Gambas – ausgelagert in eine zur Webpage gehörenden Klasse – in einer speziellen Syntax für Webpages in HTML einfügen. Die Ausgabe von (reinem) HTML übernimmt die Funktion datetime. Render() im Modul Main.module. Das ist der Inhalt des (versteckten) Ordners .scr im Projektordner:

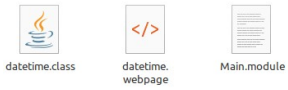

Abbildung 24.13.8.4.4: Quelltext-Dateien

Webpage – (HTML-)Datei: datetime.webpage

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>DATE.TIME</title>
    <meta charset="utf-8">
    <style>
      body {background-color:#C3DDFF;font-family:Verdana,Helvetica,font-size:32px;color:#0000FF;}
      .time-current {display: flex;margin-top: auto;}
      .time-current p {padding: 0px;font-size: 32px;text-align: left;}
    </style>
    <script>
    function startTime() {
      var today = new Date();
      var h = today.getHours();
      var m = today.getMinutes();
      var s = today.getSeconds();
      m = checkTime(m);
      s = checkTime(s);
      document.getElementById('display_current_time').innerHTML = h + ":" + m + ":" + s;
      var t = setTimeout(startTime, 1000);
    }
    function checkTime(i) {
      if (i < 10) {i = "0" + i};
      return i;
    }
    </script>
  </head>
  <body onload="startTime()">
    <br />
    Datum: <%=SetDateToGerman(Now)%><br>
    Die Webseite wurde um <%=Format$(Now, "hh:nn:ss")%> Uhr aufgerufen!
    <br>
    <time class="time-current">
      <p style="display:inline; ">Aktuelle Zeit:  </p>
      <p id="display_current_time"></p><p>  Uhr</p>
    </time>
  </body>
</html>
```

Quelltext in der Klassen-Datei der Webpage *datetime.webpage* – Gambas-Datei: datetime.class

```
' Gambas class file

Private Function SetDateToGerman(dDatum As Date) As String

  Dim aMonatMatrix, aWochenTagMatrix As New String[]
  Dim sWochenTag, sTag, sMonat, sJahr As String

  aMonatMatrix.Clear()
  aMonatMatrix = ["Januar", "Februar", "März", "April", "Mai", "Juni", "Juli", "August", "September",
                  "Oktober", "November", "Dezember"]
  aWochenTagMatrix.Clear()
  aWochenTagMatrix = Split("Sonntag Montag Dienstag Mittwoch Donnerstag Freitag Samstag", " ")

  sWochenTag = aWochenTagMatrix[WeekDay(dDatum)]
  sTag = Str(Day(dDatum))
  sMonat = aMonatMatrix[Month(dDatum) - 1]
  sJahr = Str(Year(dDatum))

  Return sWochenTag & " - " & sTag & ". " & sMonat & " " & sJahr

End
```

Startklasse – Modul-Datei: Main.module

```
' Gambas module file

Public Sub Main()
  datetime.Render()
End
```

Da Inline-CSS benutzt wird und keine weiteren Dateien wie Bilddateien o.ä. erforderlich sind, bietet es sich für einen ersten Test an, in der IDE für die Anzeige der Webseite den eingebetteten HTTP-Server zu nutzen. Dazu müssen Sie diesen HTTP-Server im Menü 'Debuggen' in der Debugger-Konfiguration 'Eingebetteten HTTP-Server benutzen' aktivieren.

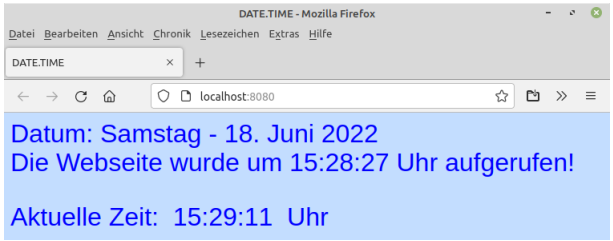

Abbildung 24.13.8.4.5: Ausgabe im eingebetteten Webserver auf Port 8080

Wenn Sie diese Ausgabe mit fortlaufender Anzeige der aktuellen Zeit sehen, dann können Sie die ausführbare Datei datetime.gambas erzeugen und unter ~/public_html/cgi-bin/datetime.gambas abspeichern.

Setzen Sie für die ausführbare Gambas-Datei datetime.gambas die angegebenen Rechte:

```
$ sudo chmod 665 $HOME/public_html/cgi-bin/datetime.gambas
```

Rufen Sie die Webseite im Webbrowser Ihrer Wahl auf:

```
http://localhost/~USERNAME/cgi-bin/datetime.gambas
```

Es sollte sich die gleiche Anzeige wie in der Abbildung 24.13.8.4.5 ergeben.

24.13.8.5 Konfiguration Modul fastcgi

Um für den Webserver die PHP-Unterstützung einzusetzen, die u.a. für Testzwecke im Zusammenhang mit der Komponente *gb.xml.rpc* und XML interessant wird, muss zusätzlich auch PHP installiert werden, wenn das noch nicht geschehen ist. Zuerst müssen Sie erkunden, welche PHP-Version auf Ihrem System installiert werden kann. Die Ausgabe des Befehls *php -v* in einer Konsole gibt einen Hinweis auf die PHP-Version, die aktuell installiert werden kann:

```
$ php -v
Der Befehl 'php' wurde nicht gefunden, kann aber installiert werden mit:
sudo apt install php7.4-cli
```

Achtung: Die Reihenfolge der Pakete sollten Sie einhalten! Ansonsten wird (automatisch) auch der Webserver `Apache2` zusätzlich installiert, was garantiert zu Problemen führen wird!

Mit dieser Befehlskette können Sie nun PHP installieren:

```
sudo apt-get install php7.4-common php7.4-cgi php7.4 php7.4-mbstring php7.4-xml php7.4-xmlrpc
```

So ermitteln Sie nach der erfolgreichen Installation die (aktuelle) PHP-Version auf Ihrem System:

```
$ php -v
PHP 7.4.3 (cli) (built: Oct  6 2020 15:47:56) ( NTS )
...
```

Das ist die Übersicht der installierten PHP-Umgebung:

```
hans@mint20:~$ dpkg --list | grep php7
ii  php7.4           7.4.3-4ubuntu2.4    all    server-side, HTML-embedded scripting language (metapackage)
ii  php7.4-cgi       7.4.3-4ubuntu2.4    amd64  server-side, HTML-embedded scripting language (CGI binary)
ii  php7.4-cli       7.4.3-4ubuntu2.4    amd64  command-line interpreter for the PHP scripting language
ii  php7.4-common    7.4.3-4ubuntu2.4    amd64  documentation, examples and common module for PHP
ii  php7.4-json      7.4.3-4ubuntu2.4    amd64  JSON module for PHP
ii  php7.4-opcache   7.4.3-4ubuntu2.4    amd64  Zend OpCache module for PHP
ii  php7.4-readline  7.4.3-4ubuntu2.4    amd64  readline module for PHP
ii  php7.4-xml       7.4.3-4ubuntu2.4    amd64  DOM, SimpleXML, XML, and XSL module for PHP
ii  php7.4-xmlrpc    7.4.3-4ubuntu2.4    amd64  XMLRPC-EPI module for PHP
```

Wenn Sie Skripte der Sprache PHP verwenden wollen, so müssen Sie auch das Modul *fastcgi* konfigurieren und aktivieren.

(1) Modul konfigurieren durch das Bearbeiten der passenden Konfigurationsdatei

Lesen Sie sich die Dokumentation zum Modul durch:

```
$ sudo gzip -d /usr/share/doc/lighttpd/fastcgi.txt.gz
```

```
$ sudo xed /usr/share/doc/lighttpd/fastcgi.txt
```

Sichern Sie das Original der Konfigurationsdatei:

```
$ sudo cp /etc/lighttpd/conf-available/10-fastcgi.conf /etc/lighttpd/conf-available/10-fastcgi.conf.old
```

Das Modul wird durch das Bearbeiten der Datei /etc/lighttpd/conf-available/10-fastcgi.conf konfiguriert:

```
$ sudo xed /etc/lighttpd/conf-available/10-fastcgi.conf
```

Ändern Sie den Inhalt der Konfigurationsdatei 10-fastcgi.conf auf diesen Inhalt:

```
## --- MODULE: FASTCGI ---
##
## Documentation: /usr/share/doc/lighttpd/fastcgi.txt
##
## Für PHP von "lighttpd" ((Fast-)CGI) muss php7-cgi oder php-cgi installiert sein!
## In /etc/php/7.4/cli/php.ini ist der Wert von 'cgi.fix_pathinfo' auf 1 zu setzen.
## For PHP of "lighttpd" ((fast)CGI, the package php7-cgi or php-cgi must be installed!
## Do not forget to set the value of cgi.fix_pathinfo to 1 in /etc/php/7.4/cli/php.ini.
##
## Aktivierung des Moduls 'fastcgi'
## Activation of the module 'fastcgi'

server.modules += ( "mod_fastcgi" )

## Fast-CGI-Server

fastcgi.server = ( ".php" => (("bin-path" => "/usr/bin/php-cgi", "socket" => "/tmp/php.sock")) )

##
## --- END OF MODULE: FASTCGI ---
```

(2) Modul aktivieren

```
$ sudo lighttpd-enable-mod fastcgi
...
Run /etc/init.d/lighttpd force-reload to enable changes
```

(3) Konfigurationsdateien neu einlesen

```
$ sudo service lighttpd force-reload
```

(4) Funktionalität des aktivierten Moduls im Webrowser testen

Die grundlegende PHP-Funktionalität testen Sie mit dem PHP-Skript ~/public_html/phpinfo.php mit folgendem Quelltext:

```
<?php phpinfo(); ?>
```

Vergessen Sie nicht, für das PHP-Skript die angegebenen Rechte zu setzen:

```
$ sudo chmod 644 $HOME/public_html/phpinfo.php
```

Rufen Sie das PHP-Skript im Webbrowser Ihrer Wahl auf:

```
http://localhost/~USERNAME/phpinfo.php
```

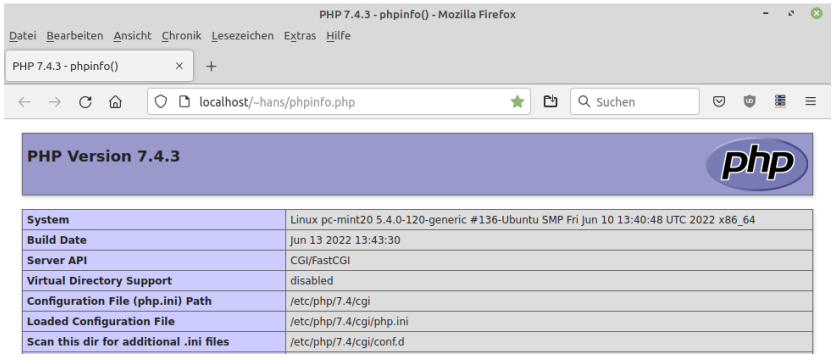

Abbildung 24.13.8.5.1: Ausgaben des PHP-Skriptes *phpinfo.php* im Webrowser

24.13.8.6 Konfiguration Modul ssi

Zum Thema 'Server Side Includes' (SSI) stand in einer Dokumentation u.a. diese Zusammenfassung:

```
SSI ist sicherlich kein Ersatz für CGI oder andere Technologien, die zur Erzeugung dynamischer
Webseiten verwendet werden. Aber es ist ein guter Weg, um kleine Mengen von dynamischen Inhal-
ten auf Seiten hinzuzufügen, ohne dabei viel zusätzliche Arbeit zu investieren.
```

Wer *Server Side Includes* in Webseiten einsetzt, wird bestätigen können: Ja, so ist es ... ! SSI ist browserunabhängig, da SSI-Anweisungen der Skript-Sprache SSI in einer Webseite vom Webserver verarbeitet werden und die Ergebnisse sofort in den HTML-Quelltext eingefügt werden, der dann zum Browser geschickt wird. Das Modul ssi ist für Sie dann interessant, wenn Sie auf komfortable Weise häufig wechselnde Texte in eine sHTML-Seite einfügen möchten.

(1) Modul konfigurieren durch das Bearbeiten der passenden Konfigurationsdatei

Lesen Sie sich die Dokumentation zum Modul durch:

```
$ sudo xed /usr/share/doc/lighttpd/ssi.txt
```

Sichern Sie das Original der Konfigurationsdatei:

```
$ sudo cp /etc/lighttpd/conf-available/10-ssi.conf /etc/lighttpd/conf-available/10-ssi.conf.old
```

Um 'Server Side Includes' nutzen zu können, müssen Sie das Modul ssi durch das Bearbeiten der Da-

tei /etc/lighttpd/conf-available/10-ssi.conf konfigurieren:

```
$ sudo xed /etc/lighttpd/conf-available/10-ssi.conf
```

Ändern Sie den Inhalt der Konfigurationsdatei 10-ssi.conf auf diesen Inhalt:

```
## --- MODUL: SSI ---
##
## Documentation: /usr/share/doc/lighttpd/ssi.txt
##
## Aktivierung des Moduls 'ssi'
## Activation of the module 'ssi'

server.modules += ( "mod_ssi" )
ssi.extension = ( ".html", ".shtml" )

## --- END OF MODULE: SSI ---
```

(2) Modul aktivieren

```
$ sudo lighttpd-enable-mod ssi
...
Run /etc/init.d/lighttpd force-reload to enable changes
```

(3) Konfigurationsdateien neu einlesen

```
$ sudo service lighttpd force-reload
```

(4) Funktionalität des aktivierten Moduls im Webbrowser testen

Zukünftig werden alle Dateien im Webordner mit der Extension *.shtml* vom Webserver Lighttpd nach SSI-Anweisungen geparst und die in ihnen enthaltenen SSI-Anweisungen ausgeführt. Beachten Sie, dass die Webserver-Option 'exec' – u.a. zum Ausführen von Systembefehlen genutzt – beim Webserver Lighttpd – offensichtlich wegen Sicherheitsbedenken – nicht implementiert wurde!

Für einen Test, ob der Webserver die Ergebnisse von SSI-Anweisungen in den HTML-Quelltext einer Webseite einbindet, wird die SHTML-Datei ssitest.shtml im Downloadbereich bereit gestellt.

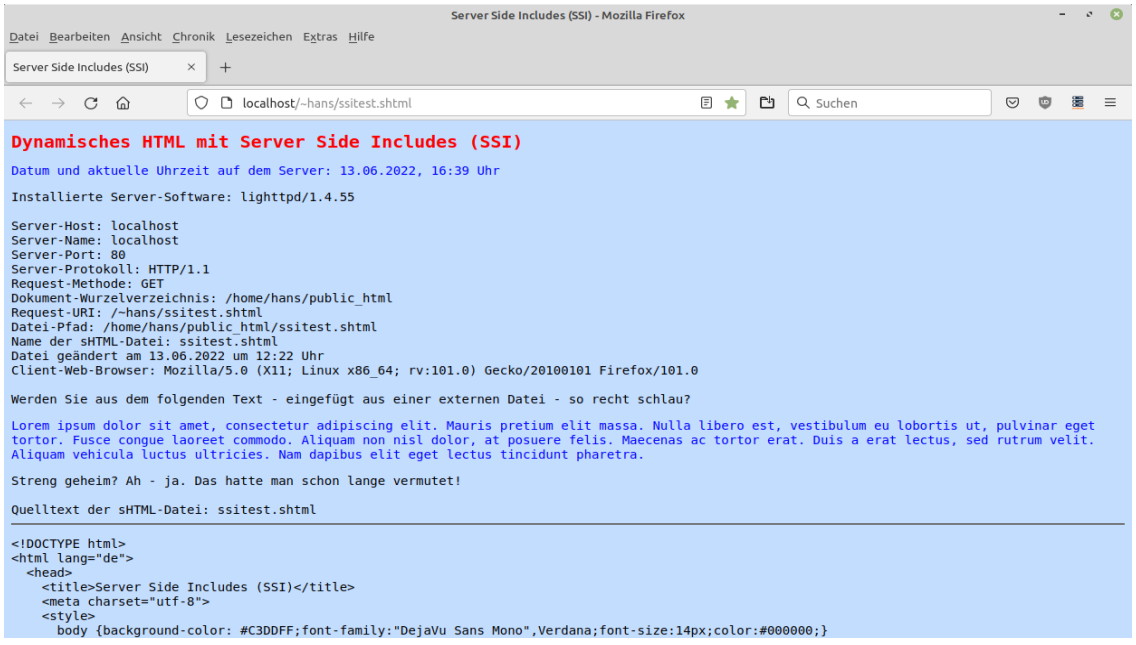

Abbildung 24.13.8.6.1: Ausschnitt der Anzeige des Inhaltes von ssitest.shtml

Die SSI-Anweisungen werden wie HTML-Kommentare in den HTML-Quelltext eingebunden. Beachten Sie, dass *<!--#SSI-Anweisung* ohne Leerzeichen zu notieren ist, jedoch das *Leerzeichen* vor dem schließenden --> notwendig ist:

```
<!--#SSI-Anweisung Attribut="Wert" -->
```

Vergessen Sie nicht, für die SSI-Datei die angegebenen Rechte zu setzen:

```
$ sudo chmod 644 $HOME/public_html/ssitest.shtml
```

Das ist der Inhalt der Datei ssitest.shtml mit Inline-CSS im Abschnitt <style>...</style>:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>Server Side Includes (SSI)</title>
    <meta charset="utf-8">
    <style>
      body {background-color: #C3DDFF;font-family:"DejaVu Sans Mono",Verdana;font-size:14px;color:#000000;}
      h1 {text-align: left; font-family:"DejaVu Sans Mono",Verdana;font-size:20px;color:#FF0000;}
      p {font-family:"DejaVu Sans Mono",Verdana;font-size:14px;color:#0000FF;}
      pre {font-family:"DejaVu Sans Mono",Verdana;font-size:14px;color:#000000;}
    </style>
  </head>
  <body>
    <h1>Dynamisches HTML mit Server Side Includes (SSI)</h1>
    <p>Datum und aktuelle Uhrzeit auf dem Server:
    <!--#config timefmt="%d.%m.%Y, %H:%M" --><!--#echo var="DATE_LOCAL" --> Uhr<br/></p>
    Installierte Server-Software: <!--#echo var="SERVER_SOFTWARE" --><br /><br />
    Server-Host: <!--#echo var="HTTP_HOST" --><br />
    Server-Name: <!--#echo var="SERVER_NAME" --><br />
    Server-Port: <!--#echo var="SERVER_PORT" --><br />
    Server-Protokoll: <!--#echo var="SERVER_PROTOCOL" --><br />
    Request-Methode: <!--#echo var="REQUEST_METHOD" --><br />
    Dokument-Wurzelverzeichnis: <!--#echo var="DOCUMENT_ROOT" --><br />
    Request-URI: <!--#echo var="DOCUMENT_URI" --><br />
    Datei-Pfad: <!--#echo var="SCRIPT_FILENAME" --><br />
    Name der sHTML-Datei: <!--#echo var="DOCUMENT_NAME" --><br />
 Datei geändert am <!--#config timefmt="%d.%m.%Y um %H:%M" --><!--#echo var="LAST_MODIFIED" -->Uhr<br />
    Client-Webbrowser: <!--#echo var="HTTP_USER_AGENT" --><br />
    <br />
    Werden Sie aus dem folgenden Text - eingefügt aus einer externen Datei - so recht schlau?
    <p><!--#include file="texte/loremipsum.html" --></p>
    Streng geheim? Ah - ja. Das hatte man schon lange vermutet!<br /><br />
    Quelltext der sHTML-Datei: <!--#echo var="DOCUMENT_NAME" -->
    <hr />
    <p><!--#include file="texte/ssitest.txt" --></p>
  </body>
</html>
```

Die SSI-Anweisungen sind im Quelltext in der Datei *ssitest.shtml* rot markiert. Die beiden einzubindenden Text-Dateien loremipsum.html und ssitest.txt im Ordner './public_html/texte' werden Ihnen auch im Downloadbereich des Kapitels zur Verfügung gestellt. So rufen Sie eine sHTML-Datei im Webbrowser auf (→ Abbildung 24.13.8.6.1):

```
http://localhost/~hans/ssitest.shtml
```

Es werden nicht nur Werte von SSI-Umgebungsvariablen ausgelesen und angezeigt, sondern auch der Text aus den o.a. zwei Text-Dateien in die generierte Website eingefügt und angezeigt.

Es lohnt sich auf jeden Fall, den von http://localhost/~hans/ssitest.shtml vom Webserver generierten HTML-Quelltext anzusehen, den Sie sich im Webbrowser Firefox mit CTRL+U anzeigen lassen und mit dem originalen SSI-Skript vergleichen können. Dann wird – wenn auch erst auf den zweiten Blick – die Funktionsweise von SSI sehr deutlich.

24.13.9 Fazit

Vom Webserver – konfiguriert in der (Basis-)Konfigurationsdatei /etc/lighttpd/lighttpd.conf und den oben beschriebenen Modulen – wird HTML-Code von statischen Webseiten und von CGI-Skripten sowie von SSI-Skripten ausgeliefert:

- Statische HTML-Webseiten *.html
- Perl-CGI-Skripte → *.pl
- Python-CGI-Skripte → *.py
- Gambas-CGI-Skripte→ *.gbw
- Gambas-CGI-Programme → Webpages → *.gambas
- PHP-Skripte → *.php
- SSI-Skripte → *.shtml

- Option: Abruf von Daten von einem XML-RPC-Server durch einen XML-RPC-Client. Hinweise zu RPC finden Sie im Kapitel 27.5.0 XML-RPC.

Alle HTML-Dateien und PHP-Dateien wie phpinfo.php sowie SSI-Skripte wie ssitest.shtml werden im Ordner ~/*public_html* abgespeichert. CGI-Skripte (*.pl, *.py, *.gbw) und CGI-Programme wie Webpages (*.gambas) gehören dagegen in den Ordner ~/*public_html/cgi-bin.*

Im Downloadbereich finden Sie in einem Archiv alle im Kapitel beschriebenen Dateien:

- test.html
- pl.pl
- py.py
- env.gbw3
- Projekt wp_datetime
- info.php
- ssitest.shtml
- Ordner 'texte' mit den Dateien loremipsum.html und ssitest.txt