

Entwicklung von Web-Applikationen mit Gambas

Kompendium von Claus Dietrich

Datum: 9.08.2022

Inhaltsverzeichnis

1 Besonderheiten von Web-Applikationen.....	1
2 Projekt-Arten für Gambas-Web-Applikationen.....	1
2.1 Web-Form-Applikationen.....	2
2.2 Web-Applikationen („Gambas Web Pages“)......	2
2.3 Web-Skript-Applikationen („Gambas Server Pages“)......	3
3 Das Common Gateway Interface (CGI).....	4
4 Besonderheiten anderer Skriptsprachen.....	8
4.1 PHP.....	8
4.2 JavaScript.....	8
4.3 Perl.....	9
4.4 Python.....	9
5 Ausführen/ Testen/ Debuggen von Web-Applikationen mit der IDE.....	10
5.1 Der Embedded Server.....	10
5.2 Automatische Übergabe an einen echten Server und Starten des Browsers.....	10
5.2.1 Bei lokalen Servern mit einem CGI-Verzeichnis unter User-Rechten.....	10
5.2.2 Bei lokalen Servern mit einem CGI-Verzeichnis unter Root-Rechten.....	12
5.2.3 Debugging von Web-Applikationen.....	13
6 Verzeichnis-Struktur für Web-Applikationen.....	13
7 Verwendung von Cascading Style Sheets (CSS).....	14
8 Installation und Einrichtung eines Lighttpd Web-Servers.....	15
8.1 Installation.....	15
8.2 Fehlersuche beim Webserver.....	17
8.3 Webserver-Steuerung.....	17
8.4 Konfiguration des Webservers.....	17
8.5 Erweiterung des Webservers.....	19
8.6 Modul CGI.....	19
8.6.1 Test mit einer Gambas Web-Applikation.....	20
8.6.2 Test mit einem Perl-Skript.....	21
8.6.3 Test mit einem Python Skript.....	21
8.6.4 Test mit einem Gambas-Skript.....	22
8.7 Modul FastCGI (PHP).....	23
8.8 Modul Userdir.....	25
8.8.1 Test mit HTML-Seiten.....	25
8.8.2 Test mit PHP-Skripten.....	26
8.8.3 Test mit CGI-Skripten.....	26
8.9 Fernzugriff auf den Lighttpd-Server.....	27
8.9.1 Übertragung von Webseiten auf den Server.....	28
9 Installation von Datenbank-Systemen für Web-Applikationen.....	30
9.1 SQLite.....	30
9.2 Datenbanksysteme mit Client-Server-Architektur.....	30

9.2.1 MySQL.....	30
10 Verwendung von Virtuellen Maschinen (VMs) für HTTP-Server.....	34
11 Internationalisierung von Gambas Web-Applikationen.....	34
11.1 Allgemein.....	34
11.2 Internationalisierung bei Gambas Web Page - Projekten.....	35
11.3 Internationalisierung von Gambas Web-Form-Applikationen.....	36
11.4 Internationalisierung von Gambas Web-Skripten.....	37

Abbildungsverzeichnis

Abbildung 1: Das CGI-Funktionsprinzip.....	5
Abbildung 2: Das Web-Formular des "Supercomputers".....	7
Abbildung 3: Das berechnete Resultat.....	7
Abbildung 4: Erzeugung der ausführbaren Datei im Server-Verzeichnis.....	11
Abbildung 5: Automatische Ausführung eines Shell-Kommandos nach Kompilierung.....	12
Abbildung 6: Verzeichnis-Struktur.....	14
Abbildung 7: Lighttpd Testseite.....	17
Abbildung 8: Ansicht Web-Form-Applikation im Debug-Modus.....	20
Abbildung 9: Ansicht Web-Form-Applikation.....	20
Abbildung 10: Anzeige im Webbrowser.....	21
Abbildung 11: Anzeige im Webbrowser.....	22
Abbildung 12: Anzeige der Umgebungsvariablen im Webbrowser.....	23
Abbildung 13: Anzeige im Webbrowser.....	25
Abbildung 14: Dialog im Datei-Manager.....	29
Abbildung 15: Web-Seite mit übersetztem Titel, Header und Footer.....	36

1 Besonderheiten von Web-Applikationen

Der Vorteil von Web-Applikationen liegt in der Tatsache, dass man zu ihrer Ausführung weder ein bestimmtes Betriebssystem, noch eine spezielle Applikation benötigt. Man verwendet auf der Nutzerseite einfach einen Web-Browser. Außerdem sind Web-Applikationen i.d.R. so ausgelegt, dass mehrere Nutzer diese gleichzeitig verwenden können.

Da sich die verschiedenen Nutzerplattformen wie, PCs, Tablets, Smartphones in ihrer Darstellungsmöglichkeiten (Bildschirmauflösung) stark unterscheiden, muss dem in einer Web-Applikation Rechnung getragen werden und das GUI sich diesen unterschiedlichen Umgebungen anpassen. Hierzu ist man gefordert, neben der grundlegenden Gestaltung per HTML, die umfassenden Formatierungsmöglichkeiten von CSS (Cascading Style Sheets) einzusetzen.

Ein weiterer Bereich, der gegenüber Desktop-Applikationen ein Umdenken erfordert, ist das Request-Response-Paradigma einer Webseite. Zwischen einem Request (Anfrage) und einer Response (Antwort) einer Webseite läuft i.d.R. nur ein kurzer einmaliger Prozess ab, der genutzt wird, um die gewünschten Informationen zu beschaffen und zur Darstellung zu bringen. Hintergrundprozesse, die kontinuierlich Daten verarbeiten, gibt es in dieser Form von Applikationen nicht. Falls erforderlich müssen separate Applikationen für solche Prozesse sorgen und der Web-Applikation die daraus gewonnenen Daten innerhalb des kurzen Request-Response-Intervalls zur Verfügung stellen.

Bei der Entwicklung von Gambas-Web-Applikationen in der IDE ist besonders auf die korrekte Verwendung von Pfaden bei HTML-Statements zu achten. So kann es z.B. vorkommen, dass eine Applikation mit folgendem Statement in der Entwicklungsumgebung mit dem embedded Server einwandfrei arbeitet und das referenzierte Icon, das sich im Projektverzeichnis „Public“ befindet, angezeigt wird:

```
""
```

Hier wurde der Pfad auch gleich in Anführungszeichen gesetzt um Probleme durch Leerzeichen im Pfad zu vermeiden.

2 Projekt-Arten für Gambas-Web-Applikationen

Gambas ist für die Entwicklung von dynamischen Webseiten (Web-Applikationen) über die CGI-Schnittstelle (Common Gateway Interface) geeignet und stellt dafür drei alternative Projekt-Arten zur Verfügung.

- Web-Form-Applikation
- Web-Applikation
- Web-Skript-Applikation

2.1 Web-Form-Applikationen

Die Projekt-Art Web-Form-Applikationen ermöglicht es, unter Verwendung der Komponente `gb.web.gui`, Applikationen in ähnlicher Form wie normale GUI-Applikationen in der IDE zu entwickeln. Neben einem `WebTimer` werden Forms und GUI-Elementen zur Verfügung gestellt, die in ähnlicher Form wie bei normalen GUI-Applikationen verwendet werden können, allerdings muss man sich bei der Festlegung von Eigenschaften der GUI-Elemente an die Erfordernisse von Webseiten anpassen. Die Festlegung von Eigenschaften der GUI-Elemente weicht deshalb auch von der normaler GUI-Applikationen ab. Bei dieser Form der Entwicklung kann zwar die Komponente `gb.web.form` alternativ zu `gb.web.gui` verwendet werden, allerdings raten die Entwickler mittlerweile davon ab.

Web-Form-Applikationen setzen im Hintergrund JavaScript ein. Damit sie funktionieren muss JavaScript auf dem Client installiert und im Web Browser aktiviert sein.

2.2 Web-Applikationen („Gambas Web Pages“)

Die Projekt-Art Web-Applikationen (Gambas Web Pages) ist seit der Gambas Version 3.1.0 verfügbar und wird ebenfalls innerhalb der IDE bearbeitet. Im Gegensatz zu Web-Form-Applikationen werden in dieser Projektart kein `WebTimer` und auch keine GUI-Elemente zur Verfügung gestellt. Hier ist der Entwickler weitestgehend auf gute HTML- und CSS-Kenntnisse angewiesen und muss sich die Bedienoberfläche damit gestalten. Damit stehen alle gewohnten Freiheiten und Möglichkeiten, die HTML bietet, zur Verfügung. Das kann auf der anderen Seite den Einsatz von JavaScript erfordern, wenn z.B. Timer-Funktionen erforderlich sind.

Die Bearbeitung des Quelltextes in der IDE hat u.a. den Vorteil, dass die Gambas-spezifischen Elemente des Quelltextes farblich gekennzeichnet sind und so eine bessere Übersicht geschaffen wird. Zur Entwicklung von komplexeren Web-Anwendungen ist diese Form der Entwicklung gegenüber der von „Gambas Server Pages“ unbedingt zu empfehlen.

Wie auch bei Web-Form-Applikationen wird diese Form der Projekte vor ihrem Einsatz auf einem Server kompiliert und mit der Dateiendung `*.gambas` versehen.

Eine Gambas Web Page besteht aus einem HTML Web-Formular, das als `*.webpage` – Datei abgespeichert wird und einer korrespondierenden Klasse, die als `*.class` - Datei abgespeichert wird.

Die `*.webpage` wird vom Compiler durch die `Render()`-Methode an den Standard-Ausgang geschickt.

Um Gambas Quelltext in HTML einfügen zu können, stellt die Klasse `Webpage` eine Syntax zur Verfügung, die an ASP (Active Server Pages) angelehnt ist und folgende Elemente umfasst:

Syntax	Beschreibung
<code><%Code%></code>	Markierung von beliebigem Gambas-Code, der ausgeführt werden soll
<code><%=Expression%></code>	Zuweisung eines Gambas-Ausdrucks, der mit der <code>Html\$</code> -Funktion konvertiert wurde.
<code><%--Comment--%></code>	Einfügung eines Kommentars, der bei der Ausführung ignoriert wird.
<code><%/%></code>	Kurzform für <code><%=Application.Root%></code> .
<code><<WebPage [Arg1_="_Value1_" ,</code>	Einschluss anderer Webseiten in die aktuelle (ähnlich HTML Markup Syntax)

Syntax	Beschreibung
<code>_Arg2_="_Value2_" ...]>></code>	
<code><%!Arg%></code>	Einfügen eines Arguments, das an ein <code><<WebPage>></code> Markup übergeben wurde.
<code><<--CONTENTS-->></code>	Markiert die Grenzen zwischen Header und Footer einer eingebundenen Webseite
<code><</WebPage>></code>	Einfügen des Footers einer eingebundenen Webseite.

Die Der Web-spezifische Bereich der Programmierung von Web-Pages basiert hauptsächlich auf der Komponente `gb.web`, die über folgende Klassen verfügt:

Klasse	Verwendung
Application	Gibt Informationen über die CGI-Applikation zurück
CGI	Für das Management der CGI-Schnittstelle
Request	Für das Management eines HTTP Requests
Response	Für das Management einer HTTP Response
Session	Für das Management von Sessions
URL	Stellte statische Hilfs-Methoden für URL-Strings zur Verfügung
WebPage	Für die Erzeugung JSP-ähnlicher dynamischer Webseiten

Hier ein Beispiel für eine „Hello World“ Gambas Web Page, wobei der Anteil des Gambas-Quelltextes farblich gekennzeichnet wurde:

```

<%
Dim a as String = "Hello World!"
Dim b as String = "This is my first Gambas CGI program."
%>
<!-- Variable declaration must come before any HTML -->
<html>
  <head>
    <title><%= a%></title>
  </head>
  <body>
    <h2><%= b%></h2>
  </body>
</html>

```

2.3 Web-Skript-Applikationen („Gambas Server Pages“)

Gambas Web-Skript-Applikationen werden nicht in der IDE entwickelt und erfordern keine Kompilierung. Die Ausführung des Codes erfolgt „on the fly“ mittels Interpreter. Zur Entwicklung von Gambas-Web-Skript-Applikationen reicht ein einfacher Texteditor. Die abgespeicherte Skript-Datei muss die Endung „`gbw3`“ haben, sie muss auf „ausführbar“ eingestellt werden und zur Ausführung muss der `gambas3-scripter` auf der Serverplattform installiert sein.

Als Syntax stehen folgende Kommandos zur Verfügung:

Syntax	Beschreibung
<%Code%>	Markierung von beliebigem Gambas-Code, der ausgeführt werden soll.
<%=Expression%>	Zuweisung eines Gambas-Ausdrucks

Da Gambas Server Pages ebenfalls die Komponente gb.web verwendet, können auch hier u.a. die Klassen Session, Response und Request verwendet werden.

Um weitere Gambas-Komponenten in das Skript einzubinden, ist folgendes Kommando vorgesehen:

```
USE "ComponentName"
```

z.B.

```
USE "gb.xml.html"
```

Hier ein Beispiel für eine „Hello World“ Gambas Server Page:

```
#!/usr/bin/env gbw3
<%
Dim a as String = "Hello World!"
Dim b as String = "This is my first Gambas CGI program."
%>
<!-- Variable declaration must come before any HTML -->
<html>
  <head>
    <title><%= a%></title>
  </head>
  <body>
    <h2><%= b%></h2>
  </body>
</html>
```

Die erste Zeile ist ein sogenanntes Shebang oder Hash-Bang, das dem Betriebssystem mit der Zeichenfolge #! mitteilt, dass es sich um ein ausführbares Skript handelt. Das Shell-Kommando /usr/bin/env startet den gbw3-Interpreter und stellt diesem eine eigene Umgebung zur Verfügung.

3 Das Common Gateway Interface (CGI)

Das Common Gateway Interface beschreibt ein Protokoll zwischen HTML-Formularen und einem CGI-Skript, das grundsätzlich so abläuft:

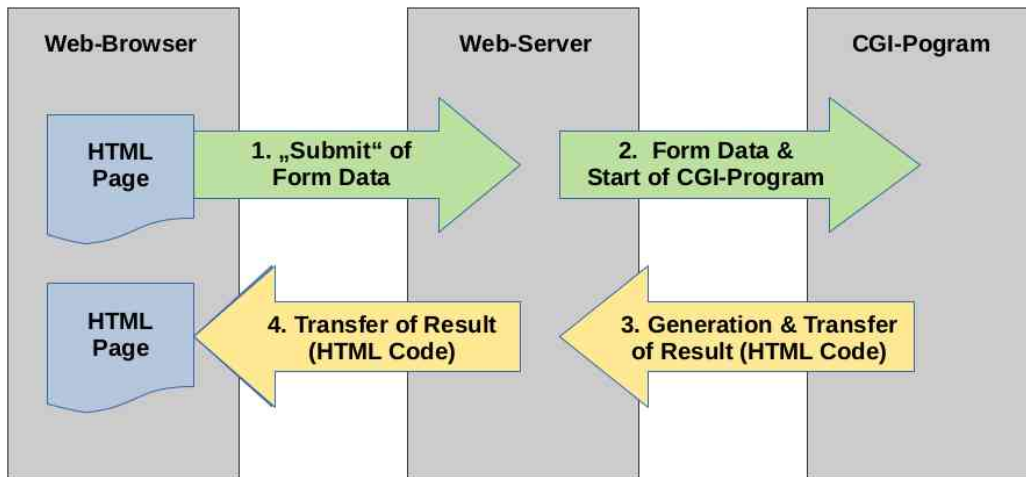


Abbildung 1: Das CGI-Funktionsprinzip

Der Client versendet eingegebene Formulare Daten entweder mit der Get- oder Post-Methode an den Server, wenn z.B. eine Taste des Typs „Submit“ angeklickt wird. Die übertragenen Formulare Daten enthalten u.a. Informationen über das Formularelement und den Wert des Formularelements. Der Server erzeugt daraufhin eine Reihe von Umgebungsvariablen und stellt sie gemeinsam mit den Formulare Daten für den Zugriff durch das CGI-Programm bereit. Daraufhin wird das CGI-Programm gestartet, dass auf Umgebungsvariablen sowie Formulare Daten zur weiteren Verarbeitung zugreifen kann. Das CGI-Programm erzeugt daraufhin als Resultat HTML-Code, der das Ergebnis in geeigneter Weise darstellt. Dieser wird an den Server geschickt, der das Ergebnis zum Client weiter reicht. Die Webseite kann auf die Weise dynamisch und interaktiv gestaltet werden.

Als Programmiersprache für CGI ist jede auf dem Server verfügbare Programmiersprache geeignet. Bevorzugt werden i.d.R. solche, mit denen die Generierung von mehrzeiligen HTML-Code einfach möglich ist.

Die Spezifikation des Common Gateway Interfaces kann hier nachgeschlagen werden:
<https://datatracker.ietf.org/doc/html/rfc3875>

Das folgende Beispiel soll die CGI-Schnittstelle demonstrieren. Dafür verwendet es eine HTML-Seite und eine Gambas Web Pages – Applikation. Die HTML-Seite enthält ein Formular, in das die Operanden eingegeben werden und die Operation ausgewählt wird. Nach dem Abschicken der Formulare Daten übernimmt die CGI.Aplikation die erzeugten Umgebungsvariablen, führt die Rechenoperation durch und erzeugt den HTML-Code zur Präsentation des Ergebnisses.

Erzeugen Sie zunächst mit einem Texteditor Ihrer Wahl eine HTML-Datei mit folgendem Inhalt und speichern Sie die Datei unter dem HTML-Verzeichnis Ihres Server unter dem Namen calc.html ab. In diesem Fall kommt das User-Verzeichnis zum Einsatz. Informationen zur Verwendung des User-Verzeichnisses erhalten Sie im Absatz „Modul Userdir“. Achten Sie auf die rot markierten Inhalte und passen sie den Usernamen und den Pfad zum CGI-Script entsprechend Ihrer Umgebung an:

```
<HTML>
<HEAD>
  <TITLE>Calculator</TITLE>
  <meta charset="UTF-8">
</HEAD>
<BODY style="font-family:Arial,Helvetica">
```



```

    h1 {text-align: left; font-family: Arial; font-size: 24px;}
  </style>
</head>
<body>
  <h1>The result is: <%= sResult%> </h1>
</body>
</html>

```

Erstellen sie dann die ausführbare Datei Namens „web_calc.gambas“ und speichern Sie diese im CGI Verzeichnis Ihrer Server-Umgebung ab (z.B. unter /home/clus/public_html/cgi-bin).

Um diesen „Web-Rechner“ zu starten, öffnen Sie jetzt die HTML-Datei (direkt oder per URL über den HTTP-Server).

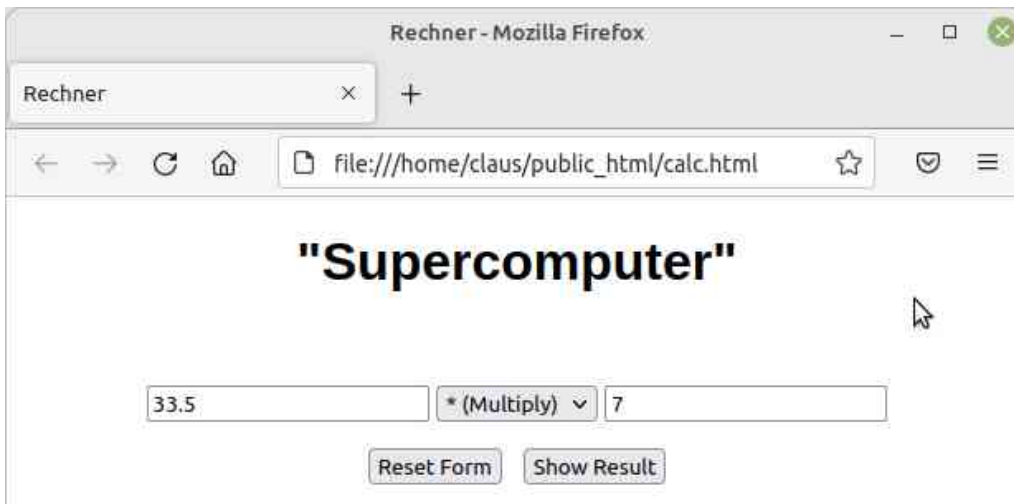


Abbildung 2: Das Web-Formular des "Supercomputers"

Geben Sie zwei Zahlen ein, wählen Sie die Rechenoperation aus und klicken Sie auf „Show Result“. Es sollte das Rechenergebnis angezeigt werden:

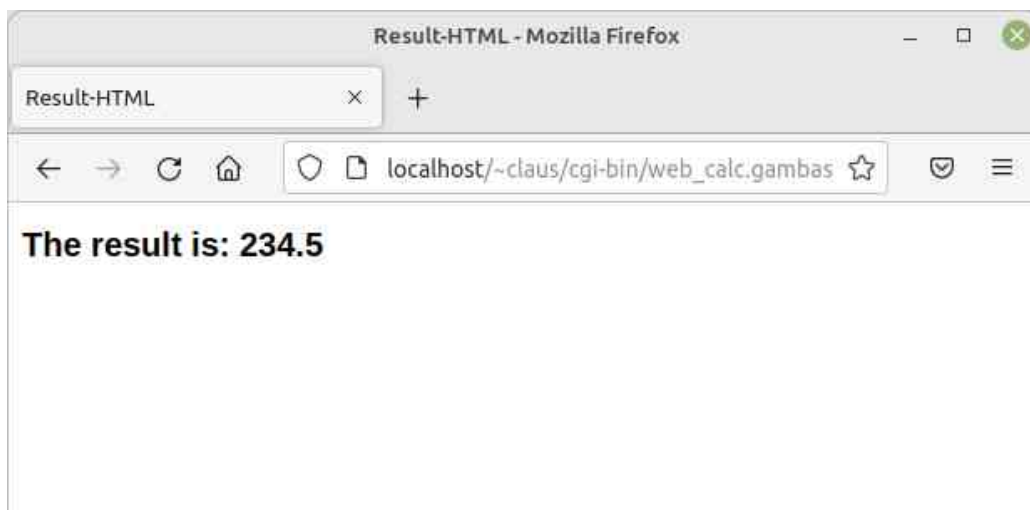


Abbildung 3: Das berechnete Resultat

Um Ergebnisse berechnen zu können ließt die Web-Applikation web_calc.gambas drei Umgebungsvariable (op, op1 und op2), die von der HTML-Webseite mit

```
<input type="submit" value="Show Result">
```

über die CGI-Schnittstelle per Post-Methode übergeben wurden. Das errechnete Ergebnis wird hier zur Demonstration auf einfache Weise mit dem Überschriften-Tag h1 angezeigt.

4 Besonderheiten anderer Skriptsprachen

4.1 PHP

PHP (rekursives Akronym für PHP: Hypertext Preprocessor) ist eine weit verbreitete Skriptsprache, die in HTML eingebettet werden kann. Das sieht typisch so aus:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php
      echo "Hello, this is a PHP-Script!";
    ?>
  </body>
</html>
```

Der PHP-Code wird also zwischen die Anfangs-Instruktion `<?php` und Abschluss-Instruktion `?>` gesetzt. Anders als JavaScript, wird PHP serverseitig ausgeführt und muss deshalb auch dort installiert sein. Für weitergehende Informationen stellt die Webseite <https://www.php.net/manual/de/intro-what-is.php> sehr hilfreiche Informationen in verschiedenen Sprachen zur Verfügung.

4.2 JavaScript

JavaScript sollte nicht mit Java verwechselt werden und ist eine der am weitesten verbreiteten Skriptsprachen. Im Gegensatz zu Java ist JavaScript eine Sprache, die auf der Client-Seite, also durch den Web-Browser durch einen Interpreter abgearbeitet wird. Das erfordert das Vorhandensein einer nutzerseitigen JavaScript-Umgebung. Diese steht für alle gängigen Betriebssysteme unter BSD-Lizenz zur Verfügung. JavaScript zeichnet sich durch eine C-ähnliche Syntax aus und ist objektorientiert.

Seit HTML 5 wird JavaScript-Code in folgender Form in HTML eingebettet:

```
<!DOCTYPE html>
<html>
<body>
  <h1>My First HTML Page</h1>
  <p>with a JavaScript Pop-Up Window.</p>
  <script>
    alert('Hello World!');
  </script>
</body>
</html>
```

Alternativ geben Sie im Bereich `<head>...</head>` den Pfad zu einer externen JavaScript-Datei an:

```
<script src="js/set_time.js" charset="utf-8"></script>
```

Diese kann sich auch im Web befinden.

Auf der Webseite https://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic_document können Sie den Code testen.

Diese Seiten bieten umfassende Informationen um JavaScript näher kennen zu lernen:

- <https://wiki.selfhtml.org/wiki/JavaScript>
- <https://javascript.info/>
- <https://www.w3schools.com/js/default.asp>

4.3 Perl

Perl ist ebenfalls eine weit verbreitete Programmiersprache, die per Interpreter abgearbeitet wird und für Web-Anwendungen serverseitig über die CGI-Schnittstelle eingesetzt wird. Der Interpreter steht unter GPL-Lizenz und ist für alle gängigen Betriebssysteme verfügbar.

Im Gegensatz zu PHP und JavaScript werden keine Perl-Skript-Blöcke in eine HTML-Seite eingebunden, um Teile der Webseite zu erzeugen. Perl-Skripte übernehmen immer die vollständige Erzeugung einer Webseite, wobei HTML-Code einfach mittels Print-Befehl wie im folgenden Beispiel ausgegeben wird.

```
#!/usr/bin/perl
print <<PAGE;
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>Hello World ...</title>
    <meta charset="utf-8">
    <style>
      body {background-color:#C3DDFF;font-family:Arial,Verdana,Courier;}
      h2 {color:blue;}
    </style>
  </head>
  <body>
    <h2>Hello World!<br />This is a very basic Perl CGI-Script.</h2>
  </body>
</html>
PAGE
```

4.4 Python

Die Programmiersprache Python wird aufgrund ihrer starken Verbreitung auf den gängigen Betriebssystemen auch gern auf Servern als Skriptsprache für das CGI-Protokoll verwendet. Die Programme werden mittels Interpreter abgearbeitet. Hier ein kleines Beispiel:

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

a = """<html>
  <head>
    <title>Hello World</title>
```

```
</head>
<body>
  <h2>Hello World! This is my first Python CGI program.</h2>
</body>
</html>"""
print (a)
```

Wer Python als Skriptsprachen näher kennen lernen möchte, kann sich hier notwendige Grundlagen erarbeiten: <https://www.w3schools.com/python/default.asp>

5 Ausführen/ Testen/ Debuggen von Web-Applikationen mit der IDE

5.1 Der Embedded Server

Zur Ausführung der Projektarten Web-Form-Applikationen und Web-Applikationen können Sie alternativ zu einem installierten Web-Server den sogenannten „Embedded Server“ verwenden“, der mit der IDE zur Verfügung gestellt wird, oder ein echten Server, den sie sich z.B. gemäß nachfolgender Beschreibung selbst installieren können. Es gilt zu beachten, dass der Embedded Server u.U. nicht alle Funktionen der Web-Applikation korrekt anzeigt. Die Verwendung eines echten Webservers ist deshalb zu bevorzugen. Die Verwendung des embedded Servers der IDE ist standardmäßig deaktiviert und muss zur Verwendung im Haupt-Menü der IDE unter „Debuggen/ Konfiguration / Debugger / eingebetteten HTTP benutzen“ aktiviert werden.

5.2 Automatische Übergabe an einen echten Server und Starten des Browsers

Tests mit echten Servern würden normalerweise so ablaufen, dass man zunächst die ausführbare Gambas-Datei erstellt, sie in des CGI-Verzeichnis des Servers kopiert, dann den Browser startet und die URL aufruft. Dieser Arbeitsablauf ist dank einer sehr nützlichen Funktion der IDE auch automatisierbar und wird wie folgt eingerichtet.

5.2.1 Bei lokalen Servern mit einem CGI-Verzeichnis unter User-Rechten

Wenn man einen lokalen Test-Server zur Verfügung hat, bei dem die CGI-Skripte unter User-Rechten gespeichert werden, kann man sich den Vorgang etwas abkürzen, indem man die ausführbare Gambas-Datei direkt im cgi-bin-Verzeichnis des Servers (z.B. /home/clus/public_html/cgi-bin) erzeugt und nicht wie üblich im Projektverzeichnis.

Öffnen sie hierzu den Dialog unter Projekt/ Erstellen/ Ausführbare Datei .../ Ort und geben dort z.B. den Pfad ein:

/home/clus/public_html/cgi-bin/webapp.gambas

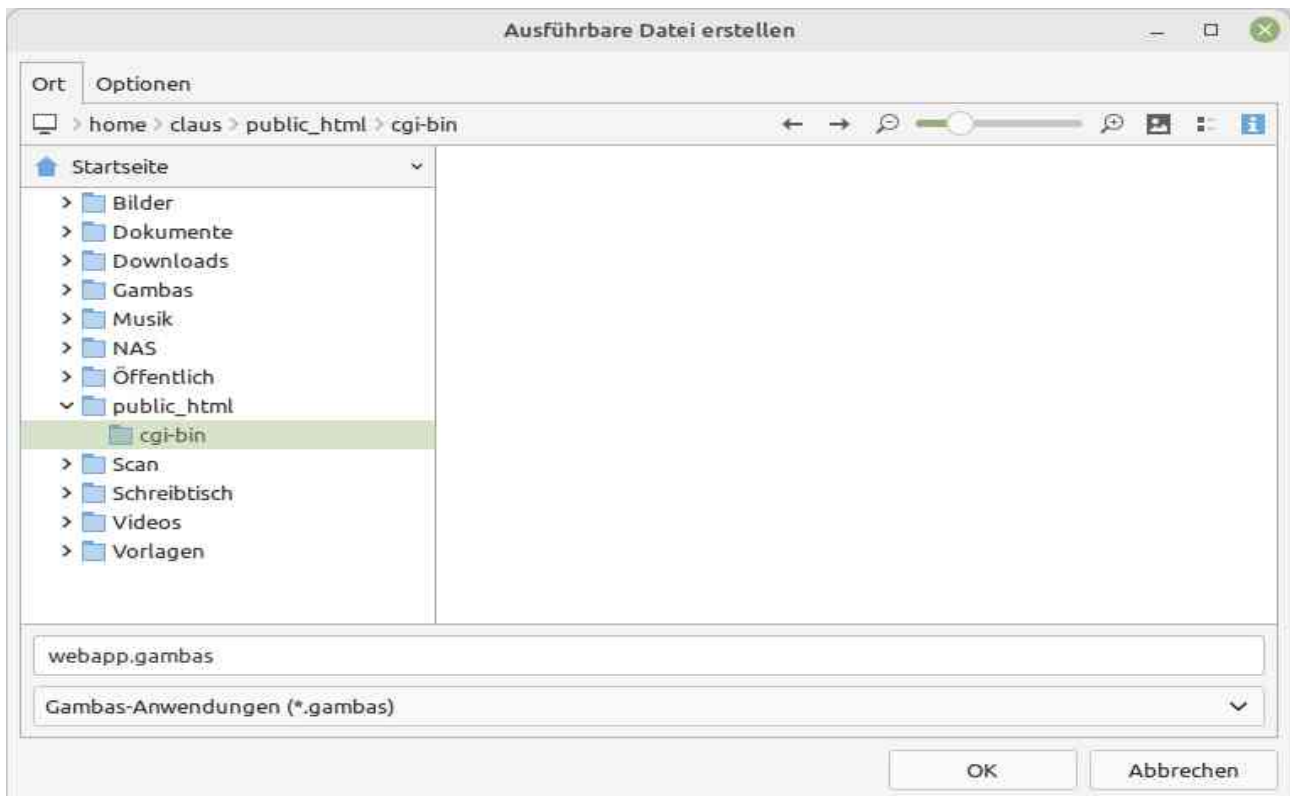


Abbildung 4: Erzeugung der ausführbaren Datei im Server-Verzeichnis

Wechseln Sie jetzt in dem selben Dialog auf den Tab „Optionen“ und geben Sie dort unter „Run this shell command after“ das folgende projektspezifische Kommando ein:

```
firefox localhost/~<UserName>/cgi-bin/<GambasAppName>
```

z.B.

```
firefox localhost/~claus>/cgi-bin/wepapp.gambas
```

Das sieht dann beispielsweise so aus:

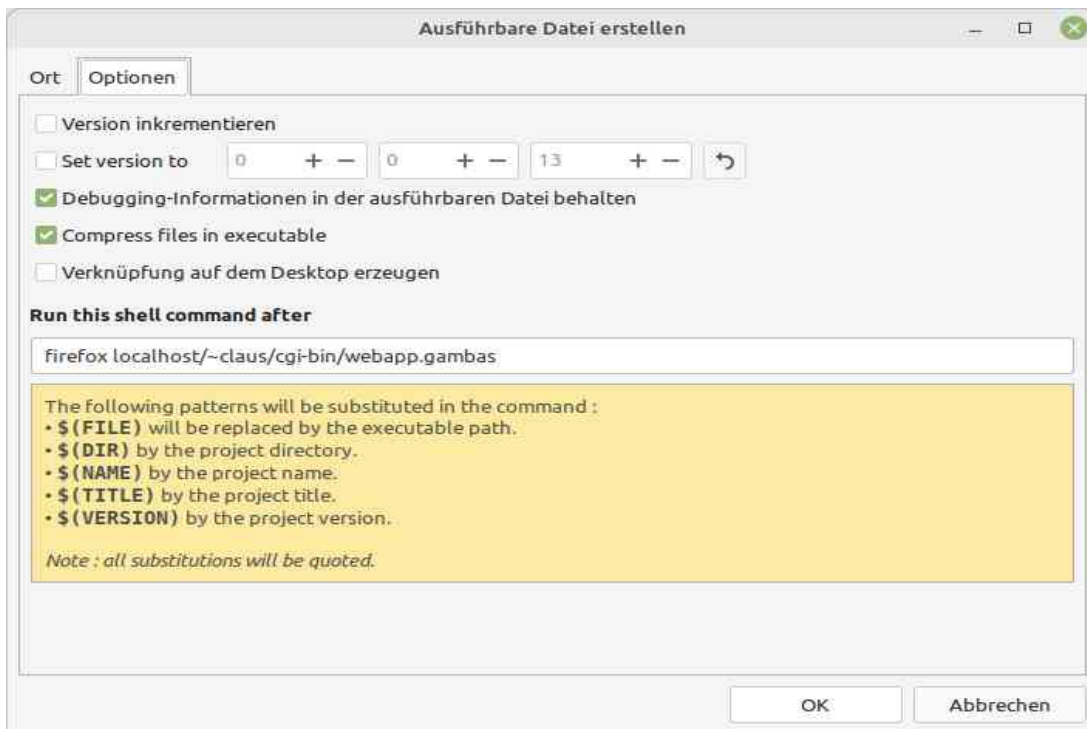


Abbildung 5: Automatische Ausführung eines Shell-Kommandos nach Kompilierung

Die IDE speichert dieses Kommando unter dem Projekt ab. Es steht danach dauerhaft zur Verfügung. Wenn Sie jetzt auf OK klicken, wird die ausführbare Gambas-Datei erzeugt und die Applikation im Firefox-Browser automatisch aufgerufen.

5.2.2 Bei lokalen Servern mit einem CGI-Verzeichnis unter Root-Rechten

Verwenden sie auf Ihrem lokalen Server für CGI-Skripte das Standard-Verzeichnis `/usr/lib/cgi-bin` das Root-Rechte erfordert, so ist auch hier nach der Erstellung der ausführbaren Gambas-Datei ein automatischer Start des Browser und Öffnen der URL möglich. Wählen Sie hierzu in dem o.a. Dialog unter dem Tab „Ort“ das übliche Projektverzeichnis zur Erzeugung der Gambas-Datei aus, also z.B.

`/home/gambas/webapp/webapp.gambas`

und geben Sie unter dem Tab „Optionen“ folgende beispielhafte Zeile in das Dialogfeld ein, wobei sie den Namen des Terminals ihrer Distribution verwenden müssen und den Pfad zur Gambas-Applikation angeben müssen:

Bei einem Cinnamon-Desktop:

```
gnome-terminal --working-directory=$(DIR) -- sudo cp $(FILE) /usr/lib/cgi-bin ; firefox localhost/cgi-bin/webapp.gambas
```

Bei einem Mate-Desktop:

```
mate-terminal --working-directory=$(DIR) -- sudo cp $(FILE) /usr/lib/cgi-bin ; firefox localhost/cgi-bin/webapp.gambas
```

Wenn Sie jetzt auf OK klicken, wird die ausführbare Gambas-Datei im normalen Projekt-Verzeichnis erzeugt und nach Eingabe des root-Passworts im Terminal wird die ausführbare Datei in das CGI-bin-Verzeichnis kopiert und im Firefox-Browser automatisch aufgerufen.

5.2.3 Debugging von Web-Applikationen

Das Debugging von Gambas Web-Applikationen muss in vielen Bereichen anders vorgenommen werden als bei Desktop-Applikationen. Folgendes hat sich in der Praxis als nützlich erwiesen:

- Validierung von HTML-Code z.B. bei https://validator.w3.org/#validate_by_input
- Validierung von CSS-Code z.B. bei <http://www.css-validator.org/validator.html.de>
- Firefox Webbrowser mit weitreichenden Entwickler-Funktionen für verschiedenste Zwecke
- Verwendung des embedded Web-Servers, besonders bei Programmierung von Gambas Web-Form-Applikationen (muss ab Version 3.17.2 aktiviert werden)
- Verwendung eines echten Web-Servers, besonders bei der Projekt-Art Gambas Web-Applikationen
- Ausgabe des HTML-Codes in der IDE-Konsole, was die Deaktivierung der Ausgabe auf den Embedded Server erfordert.

6 Verzeichnis-Struktur für Web-Applikationen

Die Verzeichnisstruktur von Webapplikation in der Gambas-IDE muss sich nicht von der anderer Gambas-Applikationen unterscheiden.

Maßgeblich für die Verzeichnisstruktur ist, dass man Web-Applikation und Daten irgendwie auf einen Server übertragen muss. Das geht natürlich am einfachsten, wenn man es mit möglichst wenig Paketen zu tun hat. Mit der Berücksichtigung folgender Regeln gelingt das für alle Formen von Web-Applikationen:

1. Alle Dateien, die mit großer Wahrscheinlichkeit nicht ausgetauscht werden müssen (z.B. Icons), werden wie üblich innerhalb des Projektordners (z.B. Public oder dessen Unterverzeichnissen) gehalten und werden damit Bestandteil der kompilierten Gambas-Datei.
2. Alle Dateien, die austauschbar sein sollen, wie z.B. Datenbanken, werden innerhalb der Projektes in dem Verzeichnis `.hidden` (oder in Unterverzeichnissen von `.hidden`) abgelegt. Dort abgelegte Dateien werden nicht Bestandteil der erzeugten ausführbaren Gambas-Datei, müssen dann aber zusätzlich mit auf den Server übertragen werden. Im Projektverzeichnis der IDE sind diese Dateien/Verzeichnisse im Verzeichnis „Projekt“ zu finden.

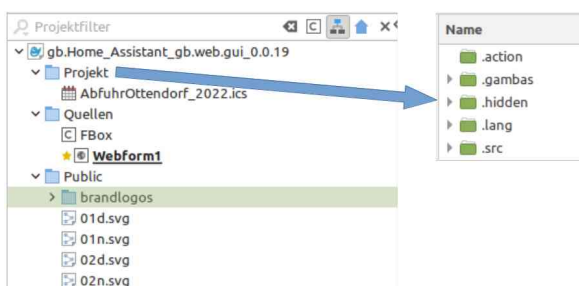


Abbildung 6: Verzeichnis-Struktur

So hat man es nur noch mit 2 Paketen zu tun, die auf einen Server übertragen werden müssen:

1. Die ausführbare Gambas-Datei und
2. das Verzeichnis .hidden auf dessen enthaltene Verzeichnisse/Daten die Web-Applikation zugreift

Da auf einem Web-Server mehrere Web-Applikation zum Einsatz kommen können, empfiehlt es sich, dort die Dateien der verschiedenen Applikationen in getrennte Ordner zu kopieren, z.B..

```
/usr/lib/cgi-bin/app_name1  
/usr/lib/cgi-bin/app_name2  
/usr/lib/cgi-bin/app_name3
```

bzw bei Ablage im Home-Verzeichnis:

```
~/public_html/cgi-bin/app_name1  
~/public_html/cgi-bin/app_name2  
~/public_html/cgi-bin/app_name3
```

Um den Kopiervorgang für die ausführbare Datei einzusparen, kann man die ausführbare Gambas-Datei bei der Kompilierung auch direkt im Verzeichnis des Servers erzeugen.

Die Übertragung der Dateien auf Server, die sich auf einer anderen Plattform, aber innerhalb des Heimnetzes befinden, unter „Fernzugriff auf den Lighttpd-Server“ beschrieben.

7 Verwendung von Cascading Style Sheets (CSS)

Bei der Entwicklung von Webapplikationen gibt es gegenüber normalen GUI-Applikationen eine Erweiterung, die gesondert beachten werden muss. Es handelt sich dabei um Cascading Style Sheets oder kurz CSS, die ein mächtiges und unverzichtbares Werkzeug zur Formatierung von Webseiten darstellen. Gambas Web-Applikationen können ebenfalls davon Gebrauch machen.

Bei der Verwendung in Web-Form-Applikationen ist es dazu erforderlich, dass man eine Datei mit Namen style.css in den Projekt-Ordner Public platziert. Damit GUI-Elemente von Web-Form-Applikationen dieses Style Sheet verwenden können, muss ihnen mitgeteilt werden welche CSS-Klasse innerhalb des Style Sheets sie verwenden sollen. Dafür ist die Eigenschaft Class der GUI-Elemente vorgesehen, in der lediglich der Name der CSS-Klasse eingetragen wird.

Bei Gambas Web-Applikationen sollte man CSS-Dateien im Projektverzeichnis Public platzieren. Die Verlinkung zu ihnen kann in HTML-Seiten mit folgender beispielhafter Syntax durchgeführt werden:

```
<link rel="stylesheet" href="style.css">
```

Zur Vereinfachung der Verlinkung kann man z.B. die Datei style.css dazu verwenden, weitere Style Sheets zu importieren. Das lässt sich beispielsweise so realisieren:

```
/* ~~~~~  
CSS  
MASTER STYLESHEET  
-----  
File name: style.css
```


The CSS file style.css is included in every HTML page:

```
<link rel="stylesheet" href="style.css">
```

```
~~~~~ */
```

```
@import url("basis.css");
@import url("header.css");
@import url("navigation.css");
@import url("table.css");
@import url("multimedia.css");
@import url("formular.css");
@import url("footer.css")
```

Das Style Sheet style.css übernimmt die Funktion des Masters und es wird nur noch zu weiteren Style Sheets verlinkt.

Achtung: Bei Web-Form-Applikationen kann man im Gegensatz zu Web-Pages-Applikationen nur eine CSS-Datei verwenden und die muss den Namen "style.css" haben. Deshalb bietet es sich bei dieser Form der Web-Programmierung mit Gambas an, die Methode des Imports anderer Style Sheets zu verwenden.

Wer die Funktionen von CSS kennenlernen möchte, der kann sich beispielsweise auf der Webseite <https://www.w3schools.com/css/> informieren.

8 Installation und Einrichtung eines Lighttpd Web-Servers

Wenn Sie Webseiten, Web-Applikationen oder ein CMS lokal testen, oder auf einer separaten Plattform im lokalen Netzwerk einsetzen wollen, dann bietet sich u.a. der leichtgewichtige und schnelle Webserver Lighttpd an. Im folgenden wird beschrieben, wie Sie ihn für diesen Zweck installieren, konfigurieren und erweitern können.

8.1 Installation

Die Installation von Lighttpd wird hier exemplarisch auf einer Mint 20.3-Distribution durchgeführt und sollte dort wie folgt vorgenommen werden. Für andere Distributionen ist mit Abweichungen zu rechnen:

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

Es empfiehlt sich, ebenfalls die Webserver-Dokumentation zu installieren:

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

Diese steht nach der Installation unter /usr/share/doc/lighttpd zur Verfügung.

Die installierte Version von Lighttpd können sie so zur Anzeige bringen:

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

Durch die Installation wird der Daemon auf Mint Distributionen automatisch gestartet. Das lässt sich durch

```
$ systemctl status lighttpd
```

oder

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

überprüfen.

Auf Debian-Distributionen wird der Daemon erfahrungsgemäß nicht automatisch gestartet, was mit

```
$ systemctl start lighttpd
```

und

```
$ systemctl enable lighttpd
```

nachgeholt werden kann. Danach steht der Server auch nach einem System-Neustart sofort zur Verfügung.

Bei der Installation wurden ein neuer user „www-data und eine neue Gruppe „www-data“ angelegt. Diesem User muss deshalb immer das Lese- und Ausführ-Recht gewährt werden.

Der Server kann unmittelbar nach der Installation mit seiner Basiskonfiguration getestet werden. Dazu ruft man im Webbrowser

```
http://localhost/index.lighttpd.html oder http://127.0.0.1/index.lighttpd.html
```

auf. Es reicht aber auch

```
http://localhost
```

weil der Server beim Aufruf automatisch nach einer „index“-Seite im Webordner sucht. Der Browser zeigt daraufhin eine vorinstallierte Webseite, die wichtige Informationen enthält.



Abbildung 7: Lighttpd Testseite

8.2 Fehlersuche beim Webserver

Den Fehlerlog des Servers können Sie sich mit dem folgenden Befehl in einer Konsole ansehen:

```
$ journalctl -u lighttpd
```

Als sehr hilfreich hat sich auch die Status-Abfrage mit

```
$ service lighttpd status
```

erwiesen, weil neben dem Status auch auf mögliche Fehlerursachen hingewiesen wird.

8.3 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
```

8.4 Konfiguration des Webservers

Vor der Durchführung von Änderungen an der Konfiguration sollte die Original-Konfigurations-Datei des Server gesichert werden:

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

Damit der Server Gambas-Web-Applikationen ausführen kann, ist an der Basis-Konfiguration lediglich der rot markierte Bereich zu ändern. Hierzu rufen sie auf:

```
$ sudo nano /etc/lighttpd/lighttpd.conf
```

Inhalt der Datei /etc/lighttpd/lighttpd.conf:

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

index-file.names        = ( "index.php", "index.html" )
url.access-deny         = ( "~", ".inc" )
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
include_shell "/usr/share/lighttpd/create-mime.conf.pl"
include "/etc/lighttpd/conf-enabled/*.conf"

#server.compat-module-load = "disable"
server.modules += (
    "mod_compress",
    "mod_dirlisting",
    "mod_staticfile",
)

```

Die Konfigurationsdatei kann nach Änderungen mit folgendem Befehl auf Fehler überprüft werden:

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

Sobald Änderungen an der Datei vorgenommen wurden, muss der Server dazu veranlasst werden, die Konfiguration neu einzulesen:

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

8.5 Erweiterung des Webservers

Zur Erweiterung des Servers stehen eine Reihe von Modulen zur Verfügung, u.a.:

- Modul cgi: CGI-Skripte der Programmiersprachen Gambas (Skripte und Webpages), Perl und Python werden im Ordner /usr/lib/cgi-bin ausgeführt
- Modul fastcgi: PHP-Skripte werden in /var/www/html ausgeführt

Die Einrichtung und Aktivierung dieser Module wird nachfolgend beschrieben. Zur Deaktivierung von Modulen können Sie jederzeit das Kommando

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

verwenden. Danach müssen Sie den Server noch einmal mit

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

veranlassen, die geänderte Konfiguration zu laden.

8.6 Modul CGI

Für die Ausführung von Gambas-, Perl- und Python-CGI-Skripten in dynamischen Webseiten ist dieses Modul erforderlich. Es sorgt dafür, dass die Skripte im Verzeichnis `/usr/lib/cgi-bin` ausgeführt werden können.

Die dazugehörige Konfigurationsdatei unter `/etc/lighttpd/conf-available/10-cgi.conf` bleibt vorerst unverändert. Falls sie dennoch geändert werden soll, sichern Sie zunächst die Original-Konfiguration des Moduls:

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

Editieren können sie die Datei dann mit

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

Das Modul wird mit

```
$
```

aktiviert und auch in diesem Fall muss der Server die veränderte Konfiguration mit

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

neu einlesen.

8.6.1 Test mit einer Gambas Web-Applikation

Testen können Sie diese neue Funktionalität, indem sie z.B. eine dynamische Web-Seite mit dem Besitzer `root` in dem Verzeichnis `/usr/lib/cgi-bin` ablegen. Erstellen Sie hierzu mit der Gambas-IDE ein neues „Web-Form-Applikation“ – Projekt und nennen Sie es `Web_Form_App`. Das neue Projekt erstellt Ihnen als Einstieg sofort eine minimale Hello-World-Applikation, welche neben Text und einer Taste die Uhrzeit als dynamisches Element anzeigt und sofort einsatzbereit ist. Aktivieren Sie im Hauptmenü der IDE unter Debuggen/ Konfiguration die Option „eingebetteten HTTP-Server benutzen“ und starten sie die Applikation. Jetzt sollten Sie folgendes im Webbrowser sehen:



Abbildung 8: Ansicht Web-Form-Applikation im Debug-Modus

Um diese Minimal-Applikation unter dem Lighttpd-Server laufen zu lassen, erzeugen Sie eine ausführbare Datei mit Namen `Web_Form_App.gambas` und kopieren sie diese mit `root`-Rechten nach `/usr/lib/cgi-bin`, z.B.:

```
$ sudo cp ~/Gambas/Web_Form_App/Web_Form_App.gambas /usr/lib/cgi-bin
```

Starten sie jetzt den Browser und geben sie als URL

```
localhost/cgi-bin/Web_Form_App.gambas
```

ein. Das Ergebnis sollte so aussehen:

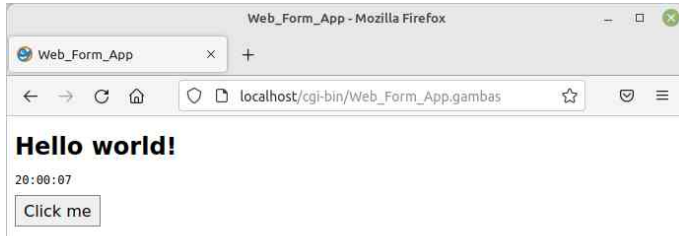


Abbildung 9: Ansicht Web-Form-Applikation

Hierbei muss besonders beachtet werden, dass die Gambas-IDE die erzeugte Gambas-Datei mit den Zugriffsrechten 755 (oktal) versieht, was im Zielverzeichnis /usr/lib/cgi-bin des Web-Servers in unveränderter Form funktioniert. Das Zugriffsrecht bedeutet rwx-rx-rx, oder in Worten:

Besitzer: lesen, schreiben, **ausführen**

Gruppe: lesen, **ausführen**

Andere: lesen, **ausführen**

wobei „ausführen“ für alle CGI-Skripte unverzichtbar ist.

8.6.2 Test mit einem Perl-Skript:

Legen sie dazu eine Datei mit

```
$ sudo nano /usr/lib/cgi-bin/pl.pl
```

an und fügen Sie folgenden Programmcode ein:

```
#!/usr/bin/perl
print "<!DOCTYPE html>";
print "<html lang=\"de\">";
print "    <head>";
print "        <title>Hello World ...</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>Hello World!<br />This is a very basic Perl CGI-Script.</h2>";
print "    </body>";
print "</html>"
```

Anders als bei einer erzeugten ausführbaren Gambas-Datei wird eine Text-Datei bei ihrer Erstellung nicht mit den notwendigen Zugriffsrechten 755 (oktal) ausgestattet. Deshalb müssen diese z.B. im Zielverzeichnis noch entsprechend angepasst werden, was mit

```
$ sudo chmod 755 /usr/lib/cgi-bin/pl.pl
```

schnell erledigt ist. Im Web-Browser wird diese Webseite mit

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

aufgerufen. Das Ergebnis sollte so aussehen:



Abbildung 10: Anzeige im Webbrowser

8.6.3 Test mit einem Python Skript

Legen sie dazu eine Datei mit

```
$ sudo nano /usr/lib/cgi-bin/py.py
```

an und fügen Sie folgenden Programmcode ein:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# Import modules for CGI handling
import cgi, cgitb
print ("<!DOCTYPE html>")
print ("<html lang=\"de\">")
print ("<head>")
print ("<title>Hello World ...</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>Hello World!<br />This is a very basic Python CGI-Script.</h2>")
print ("</body>")
print ("</html>")
```

Wie auch beim Perl-Skript müssen hier die Zugriffsrechte gesetzt werden, was mit

```
$ sudo chmod 755 /usr/lib/cgi-bin/py.py
```

schnell getan ist. Im Web-Browser wird diese Webseite so aufgerufen

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

und das Ergebnis sollte so aussehen:



Abbildung 11: Anzeige im Webbrowser

8.6.4 Test mit einem Gambas-Skript

Für diesen Test muss das Gambas-Paket gambas3-scripter installiert sein, was bei Bedarf mit

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

durchgeführt werden kann.

Legen sie eine Datei env.gbww3 in /usr/lib/cgi-bin/ mit

```
$ sudo nano /usr/lib/cgi-bin/env.gbww3
```

an und fügen Sie folgenden Gambas-Quelltext ein:

```
#!/usr/bin/env gbww3
<%
DIM sEnv AS String
'Further Gambas Code ...
%>
<!-- Variable declaration must come before any HTML -->
<html>
<h2>CGI Script Environmental Variables</h2>
<table border="1" cellspacing="0" cellpadding="2">
  <tr>
    <th align="left">Name</th>
    <th align="left">Value</th>
  </tr>
  <% FOR EACH sEnv IN Application.Env %>
    <tr valign="top">
      <td><%= sEnv %></td>
      <td><%= Application.Env[sEnv] %>&nbsp;</td>
    </tr>
  <% NEXT %>
</table>
</html>
```

Wie auch bei den anderen Skripten müssen hier die Zugriffsrechte passend gesetzt werden:

```
$ sudo chmod 755 /usr/lib/cgi-bin/env.gbww3
```

Im Web-Browser wird diese Webseite mit

```
http://localhost/cgi-bin/env.gbww3
```

aufgerufen und sieht wie folgt aus:

The screenshot shows a Mozilla Firefox browser window with the address bar displaying 'localhost/cgi-bin/env.gbw3'. The page title is 'CGI Script Environmental Variables'. Below the title is a table with two columns: 'Name' and 'Value'.

Name	Value
CONTENT_LENGTH	0
QUERY_STRING	
REQUEST_URI	/cgi-bin/env.gbw3
REDIRECT_STATUS	200
SCRIPT_NAME	/cgi-bin/env.gbw3
SCRIPT_FILENAME	/usr/lib/cgi-bin/env.gbw3
DOCUMENT_ROOT	/usr/lib/cgi-bin/
REQUEST_METHOD	GET
SERVER_PROTOCOL	HTTP/1.1
SERVER_SOFTWARE	lighttpd/1.4.55
GATEWAY_INTERFACE	CGI/1.1

Abbildung 12: Anzeige der Umgebungsvariablen im Webbrowser

8.7 Modul FastCGI (PHP)

Für die Ausführung von PHP-Skripten (im Pfad /usr/lib/) müssen die Pakete php7.4-cgi und php7.4-cli installiert sein, was Sie bei Bedarf mit

```
$ sudo apt install php7.4-common php7.4-cgi php7.4
```

sicherstellen können. Die zugehörige Konfigurationsdatei unter /etc/lighttpd/conf-available/10-fastcgi.conf bleibt unverändert. Falls sie dennoch geändert werden soll, sichern Sie zunächst die Original-Konfiguration des Moduls:

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

Editieren können sie die Datei dann mit

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

Danach sollten sie die Datei /etc/php/7.4/cli/php.ini mit

```
$ sudo xed /etc/php/7.4/cli/php.ini
```

öffnen und den Wert von cgi.fix_pathinfo auf 1 stellen, bzw das führende Semikolon in der betreffenden Zeile entfernen.

Das Modul wird danach wie folgt aktiviert:

```
$ sudo lighttpd-enable-mod fastcgi-php
```

Diese Aktivierung schließt das Modul fastcgi durch eine Abhängigkeit ein.

Auch hier muss der Server die veränderte Konfiguration mit

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

neu einlesen. Um diese neue Funktion des Servers zu testen, erzeugen sie mit

```
$ sudo nano /var/www/html/php.php
```

das Skript php.php und fügen folgenden PHP-Quelltext ein:

```
<?php
echo "<!DOCTYPE html>";
echo "<html lang=\"de\">";
echo "<head>";
echo "<title>Hello World ...</title>";
echo "<meta charset=\"utf-8\">";
echo "<style>";
echo "body {background-color:#C3DDFF;font-family:Arial,Verdana,Courier;}";
echo "h2 {color:blue;}";
echo "</style>";
echo "</head>";
echo "<body>";
echo "<h2>Hello World!<br />This is a very basic PHP-Script.</h2>";
echo "</body>";
echo "</html>"
?>
```

Im Gegensatz zu anderen Skript-Arten müssen die Zugriffsrechte von PHP-Skripten nicht auf „ausführbar“ gestellt werden. Im Web-Browser wird diese Webseite mit

```
http://localhost/php.php
```

aufgerufen und sieht wie folgt aus:

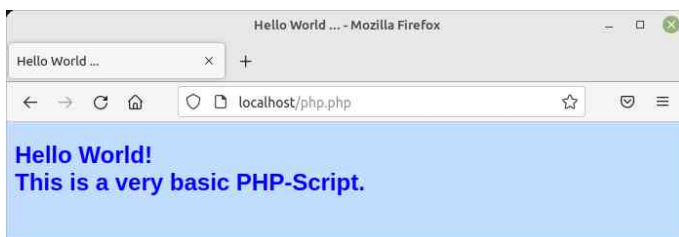


Abbildung 13: Anzeige im Webbrowser

8.8 Modul Userdir

Da man auf die bisher verwendeten Server-Verzeichnisse /var/www/html und /usr/lib/cgi-bin nur mit Root-Rechten zugreifen kann, wäre es eine erhebliche Vereinfachung, wenn man die Web-Seiten im eigenen Home-Verzeichnis ablegen könnte. Hierfür stellt das Modul Userdir eine geeignete Funktion zur Verfügung. Sie sorgt dafür, dass der User www-data auch auf die Webseiten im Homeverzeichnis zugreifen kann.

Zur Nutzung dieser Funktion muss zunächst das standardisierte Verzeichnis public_html im Home-Verzeichnis angelegt werden:

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

Das Modul userdir aktivieren Sie durch folgendes Kommando:

```
$ sudo lighttpd-enable-mod userdir
```

Danach muss der Server mit

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

die Konfiguration neu laden. Bitte kontrollieren Sie mit

```
$ stat -c '%a' /home/$USER
```

ob das User-Verzeichnis /home/\$USER die oktalen Rechte 755 besitzt. Seit Mint 21 sind die Rechte dieses Verzeichnisses nach Installation des Betriebssystems auf 750 gesetzt, was eine Verwendung des Userdirs verhindert. Mit

```
$ chmod 755 /home/$USER
```

ist das jedoch schnell in Ordnung gebracht.

8.8.1 Test mit HTML-Seiten

Ob diese neue Funktion jetzt zur Verfügung steht, testen Sie am einfachsten, indem sie die Datei test.html im Verzeichnis ~/public_html anlegen. Geben Sie dazu (jetzt ohne erhöhte Rechte)

```
$ nano ~/public_html/test.html
```

in die Konsole ein und füllen sie die Datei mit folgendem Inhalt:

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>TEST.HTML</title>
    <meta charset="utf-8">
    <style>
      body {background-color: #C3DDFF; font-family: Arial; font-size:10px;}
      h1 {text-align: left; font-family: Arial; font-size: 24px; color: blue;}
    </style>
  </head>
  <body>
    <h1>Hello World!<br />This is a very basic HTML test site.</h1>
  </body>
</html>
```

Rufen Sie jetzt die Webseite im Browser wie folgt in leicht veränderter Weise auf. Ersetzen Sie den Usernamen „claus“ durch Ihren eigenen:

```
http://localhost/~claus/test.html
```

Das Ergebnis sieht so aus:

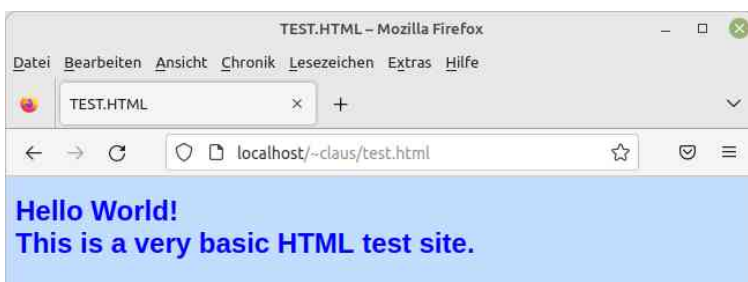


Abbildung 14: Anzeige der HTML-Seite im Webbrowser

8.8.2 Test mit PHP-Skripten

PHP-Skripte können ebenfalls in dem Verzeichnis `~/public_html` abgelegt werden. Da wir durch die vorangegangenen Tests bereits das PHP-Skript `phpinfo.php` im Standard-Verzeichnis `/var/www/html` angelegt hatten, kann dies für Testzwecke mit

```
$ sudo cp /var/www/html/php.php ~/public_html
```

ins Home-Verzeichnis kopiert und der aktuelle Besitzer „root“ mit

```
$ sudo chown claus:claus ~/public_html/php.php
```

auf den Besitzer des Home-Verzeichnisses geändert werden.

Danach kann das Skript im Browser mit

```
http://localhost/~claus/php.php
```

aufgerufen und angezeigt werden.

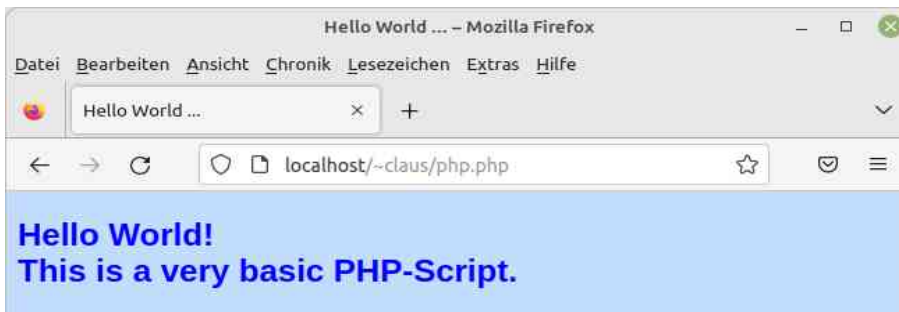


Abbildung 15: Anzeige des des PHP-Skripts im Webbrowser

8.8.3 Test mit CGI-Skripten

CGI-Web-Applikationen können sie jetzt auch im Home-Verzeichnis ablegen. Dies geschieht allerdings in einem Unterverzeichnis von `~/public_html`. Legen Sie hierzu das Unterverzeichnis `cgi-bin` an und legen Sie CGI-Web-Applikation, die sie bisher unter `/usr/lib/cgi-bin` abgelegt haben unter `~/public_html/cgi-bin` ab. Achten Sie darauf, dass als Besitzer der dort abgelegten Dateien immer der Besitzer des Home-Verzeichnisses (ihres Standard-User-Kontos) angegeben ist und dass sie über das Zugriffsrecht 755 (oktal) verfügen, was im Home-Verzeichnis der Standard ist.

Damit das auch funktioniert, muss allerdings noch folgende Änderung (rot markiert) in der CGI-Konfigurationsdatei vorgenommen werden. Dazu erstellen Sie zunächst mit

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

eine Sicherung der Originaldatei und ändern dann die Dateien `10-cgi.conf` mit einem Editor Ihrer Wahl, z.B.:

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

auf folgenden Inhalt:

```
# /usr/share/doc/lighttpd/cgi.txt

server.modules += ( "mod_cgi" )

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

cgi.execute-x-only = "enable"

cgi.assign = ( ".pl" => "/usr/bin/perl",
               ".cgi" => "/usr/bin/perl",
               ".py" => "/usr/bin/python3",
               ".gbw3" => "/usr/bin/gbw3",
               ".gambas" => "/usr/bin/gbr3" )

## Warning this represents a security risk, as it allow to execute any file
## with a .pl/.py even outside of /usr/lib/cgi-bin.
#
#cgi.assign      = (
#    ".pl" => "/usr/bin/perl",
#    ".py" => "/usr/bin/python",
#)
```

Lassen Sie dann Lighttpd mit

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

die Konfiguration neu einlesen und rufen das Perl-Skript mit

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

im Webbrowser auf. Das Ergebnisse sollte so aussehen:



Abbildung 16: Anzeige des Perl-Skripts im Webbrowser

Mit den Python-, gbw3- und Gambas-Skripten können sie analog verfahren.

Bitte beachten sie, dass Sie von nun an HTML-Seiten und CGI-Scripte sowohl in den Verzeichnissen

/var/www/HTML und

/usr/lib/cgi-bin (Besitzer = root)

als auch in

~/public_html und

~/public_html/cgi-bin (Besitzer = user)

ausführen können.

8.9 Fernzugriff auf den Lighttpd-Server

Wenn sie Lighttpd in ihrem lokalen Netzwerk auf einer separaten Plattform installiert haben, dann bietet sich besonders das SSH-Protokoll an, um Web-Seiten/Dateien einfach und sicher auf den Server zu übertragen.

Auf Klein-Rechnern wie dem Raspberry Pi ist ein SSH-Server meist schon eingerichtet und muss nur noch aktiviert werden. Ein SSH-Client ist i.d.R. auf jeder Linux-Plattform, also auch auf Ihrem Entwicklungssystem vorinstalliert.

Falls noch kein SSH-Server auf dem Web-Server vorhanden ist, so bietet sich das OpenSSH-Server-Paket an, das schnell mit

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install openssh-server
```

installiert ist. Der SSH-Server wird sofort nach der Installation und auch bei zukünftigen Neustarts des PCs automatisch gestartet. Kontrollieren können Sie das mit:

```
$ service ssh status
```

Jetzt ist es bereits möglich eine Verbindung mit dem Server aufzunehmen:

```
claus@claus-HP:~$ ssh claus@192.168.1.100
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:DIB+yYTRkuzurn7+qjJcNI0Z0jI2tveaazDqrDA38YE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts.
claus@192.168.1.100's password:
```

Beachten sie die Letzte Zeile dieses Dialogs. Sie sehen den Prompt des SSH-Servers und können jetzt per Konsole so mit ihm arbeiten, als wären Sie lokal am Server angemeldet.

Die Verbindung kann mit

```
claus@claus-VirtualBox:~$ exit
Abgemeldet
Connection to 192.168.1.100 closed.
```

wieder aufgehoben werden.

Da sich in einem lokalen Netzwerk meist ein DHCP-Server befindet, können sie die Eingabe der IP-Adresse umgehen, indem sie den Namen der Plattform angeben, z.B.:

```
claus@claus-HP:~$ ssh claus@claus-VirtualBox
The authenticity of host 'claus-virtualbox (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:DIB+yYTRkuzurn7+qjJcNI0Z0jI2tveaazDqrDA38YE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'claus-virtualbox' (ECDSA) to the list of known hosts.
claus@claus-virtualbox's password:
Last login: Sun Jul 10 10:34:13 2022 from 192.168.1.144
```

Wie Sie sehen, wird auch hier eine initiale Anerkennung des Fingerprints abgefordert.

Zum Beenden der SSH-Verbindung geben Sie `exit` ein. Danach befinden Sie sich wieder in der Konsole auf dem SSH-Client-Host. Bei allen weiteren Verbindungsversuchen vom gleichen Client verzichtet der SSH-Server auf Warnungen, da der Client den Host bei der ersten SSH-Verbindung in der Datei `~/.ssh/known_hosts` als bekannt registriert hat.

8.9.1 Übertragung von Webseiten auf den Server

Um eine Webseite auf einen Server per SSH zu übertragen bieten sich zwei Möglichkeiten an:

- per Datei-Manager
- per SSH-Kommandos in einer Konsole

Die modernen Datei-Manager von Linux-Distributionen beherrschen das SSH-Protokoll und man kann sich auf sehr einfache Weise damit bei einem SSH-Server anmelden. Hier ein beispielhafter Dialog des Datei-Managers Nemo:



Abbildung 17: Dialog im Datei-Manager

Nach der Anmeldung können Sie den Inhalt so verwalten wie gewohnt und Dateien auf den Server kopieren, als wäre es ein lokaler Pfad. Diese Form der Übertragung von Dateien ist allerdings nicht für Zielpfade geeignet, die Root-Rechte erfordern.

Natürlich können Sie Webseiten auch per Kommandozeile auf den Server übertragen. Dies geschieht (ohne existierende Verbindung zum Server) mit folgender Syntax:

```
$ scp <Pfad_quell-Datei> <SSH-user-Name>@<SSH-Server-IP>:<Pfad-SSH-Server>
```

Beispiel:

```
claus@claus-VirtualBox:~$ scp
/home/claus/Gambas/gb.Home_Assistant_gb.web.gui_0.0.18/gb.Home.gambas
claus@192.168.1.100:/home/claus/public_html/cgi-bin
claus@192.168.1.100's password:
gb.Home.gambas                               100% 875KB 30.0MB/s 00:00
```

Falls Sie ein Verzeichnis mitsamt Inhalt rekursiv auf den Server übertragen wollen, dann nutzen Sie folgende Syntax:

```
$ scp -r <Quell-Verz.> <SSH-user-Name>@<SSH-Server-IP>:<Ziel-Verz.>
```

Dies ist zum Beispiel dann erforderlich, wenn Sie das gesamte versteckte Projekt-Verzeichnis `.hidden` übertragen wollen, in dem sich alle Dateien befinden, die bei der Kompilierung nicht in die ausführbare Gambas-Datei übernommen werden und bei Bedarf auf dem Server aktualisierbar sein sollen (siehe „*Verzeichnis-Struktur für Web-Applikationen*“ für Details).

Beispiel:

```
$ scp -r /home/claus/Gambas/gb.Home2/.hidden  
claus@192.168.1.100:/home/claus/public_html/cgi-bin/gb.Home2
```

Damit ist die Webseite auf dem Server einsatzbereit. Sollten weitere Datendateien auf dem Server erforderlich sein, dann verfahren Sie mit deren Übertragung auf die gleiche Weise.

Wenn Sie bei der Übertragung die Eingabe von Passwörtern vermeiden wollen, dann bietet sich die Einrichtung einer Public-Key-Authentifizierung an, die hier im Detail beschrieben ist:

```
https://gambas-buch.de/doku.php?id=k24:k24.14:start&s[]=ssh
```

Beim Einsatz eines Servers auf einer separaten Plattform, auf den auch Dritte zugreifen können sollen, ist es natürlich wenig sinnvoll, wenn man den Aufruf von Webseiten über das User-Konto des Servers vornehmen muss (z.B. `http://192.168.1.100/~claus/cgi-bin/pl.pl`). Damit der Aufruf von Webseiten ohne Angabe eines User-Namens erfolgen kann (z.B. mit `http://192.168.1.100/cgi-bin/pl.pl`) müssen CGI-Applikationen in das Verzeichnis

```
/usr/lib/cgi-bin
```

bzw HTML-Seiten in das Verzeichnis

```
/var/www/html
```

kopiert werden, was bei bestehender SSH-Verbindung für einzelne Dateien wie folgt vorgenommen werden kann:

```
$ sudo mv ~/public_html/cgi-bin/pl.pl /usr/lib/cgi-bin
```

Der Besitzer sollte dann noch auf „root“ gesetzt werden:

```
$ sudo chown root:root /usr/lib/cgi-bin/pl.pl
```

Mit Verzeichnissen kann analog verfahren werden:

```
$ sudo mv ~/public_html/cgi-bin/.hidden /usr/lib/cgi-bin  
$ sudo chown -R root:root /usr/lib/cgi-bin/.hidden
```

9 Installation von Datenbank-Systemen für Web-Applikationen

Falls bei den Web-Applikationen Datenbanksysteme eingesetzt werden sollen, müssen die nachfolgend beschriebenen Installationen durchgeführt werden.

9.1 SQLite

SQLite ist eine beliebtes und leichtgewichtiges SQL-Datenbanksystem, das ohne Client-Server-Architektur auskommt. Im Gegensatz zu anderen Datenbanksystemen ist eine Installation von

SQLite nur dann erforderlich, wenn Sie beabsichtigen den Client für die Bearbeitung der Datenbank zu verwenden. Die Installation kann wie folgt vorgenommen werden:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install sqlite3
```

9.2 Datenbanksysteme mit Client-Server-Architektur

Für Herstellung eines Datenbanksystems mit Client-Server-Umgebung sind neben der Installation auch User und gegebenenfalls Datenbanken einzurichten.

9.2.1 MySQL

MySQL ist ein bekanntes und verbreitetes Datenbanksystem für Client-Server-Architekturen. Es steht für verschiedenste Betriebssysteme und neben einer proprietären Lizenz auch unter GNU General Public License V2 zur Verfügung. Diese Version kann unter Mint 20.3 wie folgt installiert werden:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install mysql-server
```

Bei der Installation wird auch der MySQL-Client installiert, obwohl dieser bei Linux Mint im Software-Manager als „nicht installiert“ angezeigt wird. Ob bei der Installation alles glatt gelaufen ist, können sie mit folgendem Kommando prüfen

```
$ mysql --version
```

Der Server wird beim Abschluss der Installation automatisch gestartet, was Sie wie folgt kontrollieren können:

```
$ systemctl status mysql
• mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:~>
   Active: active (running) since Mon 2022-08-01 14:54:36 CEST; 7min ago
   Main PID: 6054 (mysqld)
   Status: "Server is operational"
   Tasks: 38 (limit: 4607)
   Memory: 361.1M
   CGroup: /system.slice/mysql.service
           └─6054 /usr/sbin/mysqld

Aug 01 14:54:35 claus-VirtualBox systemd[1]: Starting MySQL Community Server...
Aug 01 14:54:36 claus-VirtualBox systemd[1]: Started MySQL Community Server.
lines 1-12/12 (END)
```

Für die Steuerung des Servers stehen folgende Optionen zur Verfügung:

```
$ sudo service mysql <start|stop|restart|reload|force-reload|status>
```

Verwenden Sie jetzt folgendes Kommando um sich einzuloggen:

```
$ sudo mysql
```

einloggen. Das Anmeldeverfahren des Roots muss in diesem Fall wie folgt geändert werden:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '<root-  
password>';  
Query OK, 0 rows affected (0,01 sec)  
  
mysql> quit
```

Ab jetzt können Sie sich nicht mehr mit `$ sudo mysql` anmelden und müssen das native MySQL-Login-Verfahren verwenden.

Geben Sie jetzt folgendes Kommando ein und führen Sie den Dialog so durch:

- Geben Sie das Root-Passwort ein, das Sie zuvor über das ALTER USER Kommando festgelegt haben
- Treffen Sie eine Auswahl über die Stärke der zu verwendenden Passwörter
- Passen Sie das Root-Passwort gegebenenfalls an die gewählte Passwort-Richtlinie an
- Beantworten Sie alle anderen Fragen vorzugsweise mit ‚y‘.

```
$ sudo mysql_secure_installation
```

Securing the MySQL server deployment.

Enter password for user root:

VALIDATE PASSWORD COMPONENT can be used to test passwords and improve security. It checks the strength of password and allows the users to set only those passwords which are secure enough. Would you like to setup VALIDATE PASSWORD component?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW Length >= 8

MEDIUM Length >= 8, numeric, mixed case, and special characters

STRONG Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
Using existing password for root.

Estimated strength of the password: 100

Change the password for root ? ((Press y|Y for Yes, any other key for No) : n

... skipping.

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at

the root password from the network.

```
Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.
```

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

```
Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
- Dropping test database...
Success.
```

```
- Removing privileges on test database...
Success.
```

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

```
Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.
```

All done!

Legen Sie jetzt einen User an und loggen Sie sich dafür mit der nativen MySQL-Methode ein:

```
$ mysql -u root -p
```

Als Passwort geben Sie bei der Abfrage das oben festgelegte "<root-password>" an.

Legen Sie jetzt einen Datenbank-User an, wobei Sie die Ausdrücke <user_name> und <user-password> ihrem Bedarf entsprechend ersetzen müssen. Der Ausdruck ,%' steht für „von jedem Host“. D.h. sowohl lokal als auch über eine lokale Netzwerkverbindung.

```
mysql> CREATE USER '<user_name>'@'%' IDENTIFIED BY '<user-password>';
Query OK, 0 rows affected (0,02 sec)
```

Danach werden dem angelegten User geeignete Privilegien eingerichtet, wobei der Ausdruck *.* vermieden werden sollte und man die Privilegien vorzugsweise auf die relevante(n) Datenbank(en) beschränken sollte:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO '<user_name>'@'%;
Query OK, 0 rows affected (0,00 sec)
```

Führen Sie danach dieses Kommando aus:

```
mysql> FLUSH PRIVILEGES;
```

und loggen Sie sich danach aus dem MySQL-Prompt mit

```
mysql> quit
```

aus. Damit der User sich tatsächlich von jedem Host im lokalen Netzwerk einloggen kann, muss jetzt noch die Konfiguration von MySQL geändert werden:

```
$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Suchen nach

```
bind-address = 127.0.0.1
```

und ändern Sie diesen Eintrag auf

```
bind-address = 0.0.0.0
```

ab. Danach starten Sie den Server neu:

```
$ sudo service mysql restart
```

Falls verfügbar, können Sie sich als Test mit dem neu angelegten User ‚claus‘ jetzt von einer anderen Plattform aus mit dem MySQL-Client einloggen:

```
$ mysql -u <user> -h 192.168.1.100 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.30-0ubuntu0.20.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> quit
```

Damit wäre das das MySQL-DBMS soweit vorbereitet. Auf die Einrichtung eines Remote Admins wird hier verzichtet und empfohlen den MySQL-root über eine SSH-Verbindung zu verwenden.

Falls Sie MySQL installieren wollen, so sollten Sie die folgende Prozedur genau einhalten:

```
$ sudo apt-get remove --purge mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-
server mysql-server-8.0 mysql-server-core-8.0
```

Danach müssen Sie folgende Verzeichnisse löschen:

```
$ sudo rm -rf /var/lib/mysql* /var/log/mysql
```

10 Verwendung von Virtuellen Maschinen (VMs) für HTTP-Server

Die Verwendung von virtuellen Maschinen als lokaler HTTP-Server benötigt zwar mehr Platz auf der Festplatte, hat sich aber als besonders praktisch erwiesen. Bei Misserfolgen kann man einfach wieder von vorn anfangen. Die User-Plattform bleibt davon unberührt. Hinzu kommt, dass man eine fertig eingerichtete VM exportieren kann und bei Bedarf innerhalb weniger Sekunden auf beliebigen Plattformen mit beliebigen Betriebssystemen wieder in Betrieb nehmen kann.

Zur Einrichtung eines HTTP-Servers geht man genau so vor wie auf einer normalen Plattform. Lediglich die IP-Adresse ist eine andere und unabhängig vom Host. Es bietet sich an, auch gleich die Gambas-IDE mit zu installieren, womit man sich den Aufwand zur Übertragung einer Webseite auf den Server minimiert. Ein Browser sollte für diesen Fall auch verfügbar sein.

Zur Einrichtung von virtuellen Maschinen bieten sich u.a. an:

- VirtualBox (kommerzielle und kostenlose Version für private Anwendung)
- QEMU + virt-manager (GPL2/GPL3))
- VMWare (kommerzielle und kostenlose Version für private Anwendung)

11 Internationalisierung von Gambas Web-Applikationen

11.1 Allgemein

Web-Applikationen, die im Internet eingesetzt werden, können in der Regel aus jeder Sprachregion heraus aufgerufen werden. Deshalb kann es von Interesse sein, dass Web-Applikationen sich auf die Sprachregion des Clients einstellen. Um das zu ermöglichen, muss der Client (Browser) den Server zunächst über die bevorzugten Sprachen informieren. Das geschieht bei Browsern automatisch bei jedem Request, der an den Server geschickt wird. Diese Information kann vom Server an die Web-Applikation weitergereicht werden. So ist es möglich, dass Web-Applikationen sich durch angepasste Formatierungen und sprachliche Übersetzungen auf Clients einstellen.

Typischerweise sind Browser auf die lokale Sprachregion mit Priorität 1 voreingestellt, allerdings ist in der Regel die us-englische Sprache (en-us) als weitere „Accept-Language“ mit niedrigerer Priorität mit voreingestellt. Diese Browser-Einstellungen lassen sich jederzeit nach Bedarf verändern.

11.2 Internationalisierung bei Gambas Web Page - Projekten

Damit sich Web Page - Projekte auf die Sprache des Clients einstellen können, bietet die Komponente gb.web eine Schnittstelle über die Request-Klasse. Die Eigenschaft Request.Language beinhaltet die erforderliche Information und wird in folgendem Beispiel dazu genutzt, die Gambas-Applikation auf die Sprache des Clients einzustellen. Das Zahlen- und Datums-Format wird durch die Zeile „System.Language = Request.Language“ automatisch angepasst. Wenn die Web-Applikation in der IDE zusätzlich auf „übersetzbar“ eingestellt wird und die Begriffe in Klammern in der IDE übersetzt werden, werden diese in übersetzter Form im Browser dargestellt:

```
<%  
Dim sEnv As String  
System.Language = Request.Language  
%>  
  
<html>  
<body>  
  <h1><%=("Gambas Web Application")%></h1>  
  <h2><%=("Current Date : ") & Format(Now, "dd. mmmm yyyy")%></h2>  
  <h3><%=("Number format : ") & Format(1.234, "#.000")%></h3>  
  <table border="1" cellpadding="4" cellspacing="0">  
    <tr>  
      <th>Variable</th>  
      <th>Value</th>  
    </tr>  
  </table>  
<%  
For Each sEnv In Env  
%>
```

```

        <tr>
            <td><%=sEnv%></td>
            <td><%=Env[sEnv]%></td>
        </tr>
    <%
    Next
    %>
</table>
</body>
</html>

```

Das Einfügen von Gambas-Ausdrücken funktioniert auch innerhalb einer Gambas-Include-Instruktion, wie das folgende Beispiel zeigt. Es handelt sich um die Webseite „Page“, die in eine andere Webseite eingebettet wird. Zur Übersetzung des Titels wird anstatt des Titel-Strings ein Gambas-Ausdruck übergeben:

Webseite Main:

```

<%
System.Language = Request.Language
%>
<<IncludePage title=<%=("Gambas WebPage Example")%> >>
    <h2><%Print ("Gambas WebPages are easy!")%></h2>
    <h2><%Print ("Time : ")%><%=Format(Now, "hh:nn:ss")%></h2>
<</IncludePage>>

```

Das ist der Inhalt der eingebetteten Webseite „IncludePage“:

```

<!DOCTYPE html>

<html>

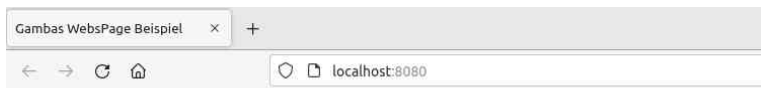
<head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <link rel="stylesheet" href="<%/>/style.css">
    <link rel="icon" href="<%/>/img/icon.png" type="image/png">
    <title><%!title%></title>
</head>

<body>
<h2><%=("Sample header of the included web-page.")%></h2>
<<- - CONTENTS - ->>
<h2><%=("Sample footer of the included web-page.")%></h2>
</body>

</html>

```

Hier das Ergebnis der übersetzten Webseiten:



Beispiel eines Headers einer eingebetteten Webseite.

Gambas WebPages sind leicht!

Uhrzeit : 21:37:39

Beispiel eines Footers einer eingebetteten Webseite.

Abbildung 18: Web-Seite mit übersetztem Titel, Header und Footer

11.3 Internationalisierung von Gambas Web-Form-Applikationen

Die Internationalisierung von Web-Form-Applikationen kann in gleicher Form vorgenommen werden wie bei Nicht-Web-Applikationen. Datums- und Zahlenformate passen sich automatisch an die Sprache des anfragenden Browsers an und Begriffe in Klammern können wie üblich übersetzt werden, sofern das Projekt auf „übersetzbar“ eingestellt wurde.

11.4 Internationalisierung von Gambas Web-Skripten

Zur Internationalisierung von Gambas Web-Skripten kann wie bei Gambas Web-Page-Projekten vorgegangen werden. Hier ein Beispiel:

```
#!/usr/bin/env gbw3
<%
DIM sEnv AS String
System.Language = Request.Language
'Further Gambas Code ...
%>
```

Die Zeile „System.Language = Request.Language“ sorgt auch hier für korrekte Datums- und Zahlenformate. Mechanismen zur Übersetzung von Begriffen sind bei Gambas Web-Skripten allerdings nicht verfügbar. Bei überschaubarem Bedarf und Aufwand, ließe sich hierfür eine geeignete Routinen schreiben, allerdings sollte man abwägen, ob die Auswahl eines anderen Projekttyps sinnvoller ist.