24.6.10 Entwicklung von Web-Applikationen mit Gambas

Inhaltsverzeichnis

24.6.10 Entwicklung von Web-Applikationen mit Gambas....1
24.6.10.1 Besonderheiten von Web-Applikationen....1
24.6.10.2 Projekt-Arten....2
24.6.10.2.1 Web-Form-Applikationen....2
24.6.10.2.2 Web-Applikationen....2
24.6.10.2.3 Web-Skript-Applikationen....3
24.6.10.3 Das Common Gateway Interface (CGI)....4
24.6.10.4 Besonderheiten der verschiedenen Skriptsprachen für dynamische Webseiten....6
24.6.10.4.1 PHP....6
24.6.10.4.2 JavaScript....6
24.6.10.4.3 Perl....7
24.6.10.4.4 Python....7
24.6.10.5 Ausführen und Testen von Gambas Web-Applikationen....8
24.6.10.5.1 Der Embedded-Webserver....8
24.6.10.5.2 Automatische Übergabe an einen echten Server und Starten des Browsers....8
24.6.10.6 Verzeichnis-Struktur für Web-Applikationen....10
24.6.10.7 Verwendung von Cascading Style Sheets (CSS)....10
24.6.10.8 Installation und Einrichtung eines Lighttpd Webservers....11
24.6.10.8.1 Installation....11
24.6.10.8.2 Fehlersuche beim Webserver....12
24.6.10.8.3 Webserver-Steuerung....12
24.6.10.8.4 Konfiguration des Webservers....13
24.6.10.9 Erweiterung des Webservers....14
24.6.10.9.1 Modul CGI....14
24.6.10.9.2 Modul FastCGI (PHP)....18
24.6.10.9.3 Modul UserDir....19
24.6.10.9.4 Modul SSI....21
24.6.10.10 Fernzugriff auf den Webserver Lighttpd....23
24.6.10.10.1 Übertragung von Webseiten auf den Server....24
24.6.10.11 Installation von Datenbank-Systemen für Web-Applikationen....26
24.6.10.12 Verwendung von 'Virtuellen Maschinen (VMs)' für HTTP-Server....26

24.6.10.1 Besonderheiten von Web-Applikationen

Der Vorteil von Web-Applikationen liegt darin, dass man zu ihrer Ausführung weder ein bestimmtes Betriebssystem, noch eine spezielle Applikation erforderlich ist. Man benötigt auf der Nutzerseite nur einen Webbrowser. Außerdem sind Web-Applikationen in der Regel so ausgelegt, dass mehrere Nutzer diese gleichzeitig verwenden können.

Da sich die verschiedenen Nutzerplattformen wie PCs, Tablets oder Mobiltelefone in ihren 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 mit 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 nur ein kurzer einmaliger Prozess ab, der genutzt wird, um die gewünschten Informationen zu beschaffen und darzustellen. Hintergrundprozesse, die kontinuierlich Daten verarbeiten, gibt es bei 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 den HTML-Elementen zu achten. So kann es zum Beispiel vorkommen, dass eine Applikation mit folgendem HTML-Element in der Entwicklungsumgebung mit dem Embedded-Webserver einwandfrei arbeitet und das referenzierte Icon, das sich im Projektverzeichnis "Public" befindet, angezeigt wird:

```
"<img src=“icon.svg height='32px' width='32px'>"
```

Wenn Sie die Applikation dann aber kompiliert haben und sie auf einem Server startet, so wird das Icon nicht angezeigt. Um die Welten von Entwicklungsumgebung mit der eines realen Webservers in Einklang zu bringen, sollte man es sich zur Regel machen, innerhalb eines HTML-Elements das CGI-Root-Verzeichnis mit zu verwenden, was dann zum Beispiel so aussieht:

```
"<img src='" & Application.Root &/ "icon.svg" & "' height='32px' width='32px'>"
```

Hier wurde der Pfad zusätzlich in Anführungszeichen gesetzt, um Probleme durch Leerzeichen im Pfad zu vermeiden.

24.6.10.2 Projekt-Arten

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.

24.6.10.2.1 Web-Form-Applikationen

Die Projekt-Art Web-Form-Applikationen ermöglicht es, unter Verwendung der Komponente gb.web.gui, Applikationen in ähnlicher Weise wie normale GUI-Applikationen in der IDE zu entwickeln. Neben einem Web-Timer stehen auch hier Forms und GUI-Elemente zur Verfügung. Allerdings muss man sich bei der Festlegung von Eigenschaften der GUI-Elemente an die Erfordernisse von Webseiten anpassen. Bei dieser Art der Projekte 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 auch JavaScript ein. Damit sie funktionieren, muss JavaScript auf dem Client installiert und im Webbrowser aktiviert sein.

24.6.10.2.2 Web-Applikationen

Die Projekt-Art Web-Applikation 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 Timer und auch keine GUI-Elemente zur Verfügung gestellt. Hier ist der Entwickler auf gute HTML-Kenntnisse angewiesen und muss sich die Bedienoberfläche damit gestalten. Auf diese Weise erhält sich ein Entwickler einer Webseite alle gewohnten Freiheiten und Möglichkeiten, die HTML bietet. Das kann auf der anderen Seite aber den Einsatz von JavaScript erfordern, wie bei der Implementierung von Timer-Funktionen. 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 Art der Projekte vor ihrem Einsatz auf einem Server kompiliert und mit der Dateiendung *.gambas versehen. Eine Gambas-Webpage besteht aus einem HTML Web-Formular, das als *.webpage-Datei abgespeichert wird und einer korrespondierenden Klasse, die als *.class-Datei abgespeichert wird. Durch die Render()-Methode wird die gerenderte *.webpage-Datei vom Compiler an den Standard-Ausgang geschickt.

Die Klasse Webpage verwendet eine spezielle Syntax – die an ASP (Active Server Pages) angelehnt ist – um Gambas-Quelltext in HTML einzufügen.

Syntax	Beschreibung
<%Quelltext%>	Jeder Gambas-Quelltext wird 'so wie er ist' ausgeführt.
<%=Expression%>	Zuweisung eines Gambas-Ausdrucks, der mit der Html$-Funktion konvertiert wurde.
<%--Comment--%>	Dieser Webpage-Kommentar wird vollständig ignoriert, während ein

Syntax	Beschreibung
	HTML-Kommentar wie '<!-- HTML-COMMENT -->' im HTML ausgegeben wird.
<%/%>	Eine Abkürzung für <%=Application.Root%>.
<<Webpage>>	Fügt den Inhalt einer anderen Webseite in die aktuelle Webseite ein.
<<Webpage arg1="value1" arg2="value2" … >>	Fügt den Inhalt einer anderen Webseite mit (optionalen) Attributen in die aktuelle Webseite ein. Beispiel: <<AstroWebpage name="sterne" anzahl="7">>
<%!arg_name%>	Liefert den Wert des Attributs ´arg_name.´
<<--CONTENTS-->>	Kennzeichnet die Grenze zwischen der Kopf- und der Fußzeile innerhalb der eingebundenen Webpage.
<</Webpage>>	Einfügen des Footers einer eingebundenen Webseite.

Tabelle 24.6.10.2.1 : Webpage-Syntax

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	Stellt statische Methoden für URL-Strings zur Verfügung
Webpage	Für die Erzeugung JSP-ähnlicher dynamischer Webseiten

Tabelle 24.6.10.2.2 : Klassen in der Komponente gb.web

Hier ein Beispiel für eine "Hello World"-Gambas-Webpage – einem Mix aus HTML und Gambas-Quelltext:

```
<%
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>
```

24.6.10.2.3 Web-Skript-Applikationen

Web-Skript-Applikationen werden nicht in der IDE entwickelt und erfordern keine Kompilierung. Die Ausführung des Codes erfolgt 'on the fly' durch den Interpreter. Zur Entwicklung von Web-Skript-Applikationen reicht ein einfacher Texteditor. Die abgespeicherte Skript-Datei muss die Endung 'gbw3' haben und sie muss auf 'ausführbar' eingestellt werden. Zur Ausführung muss der Gambas3-Scripter auf der Serverplattform installiert sein. Unter https://gambaswiki.org/wiki/doc/serverpage können Sie Grundlagen zu den 'Gambas Server Pages' nachlesen.

Syntax	Beschreibung
<%Quelltext%>	Jeder Gambas-Quelltext wird 'so wie er ist' ausgeführt.
<%=Expression%>	Zuweisung eines Gambas-Ausdrucks.

Tabelle 24.6.10.2.3 : Skript-Syntax

Da Gambas Server Pages ebenfalls die Komponente gb.web verwenden, können auch hier u.a. die Klassen Session, Response und Request verwendet werden. Um weitere Gambas-Komponenten in das Skript einzubinden, ist folgendes Kommando im Skript vorgesehen:

```
USE "ComponentName"
Beispiel:
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 Hashbang, 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.

24.6.10.3 Das Common Gateway Interface (CGI)

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

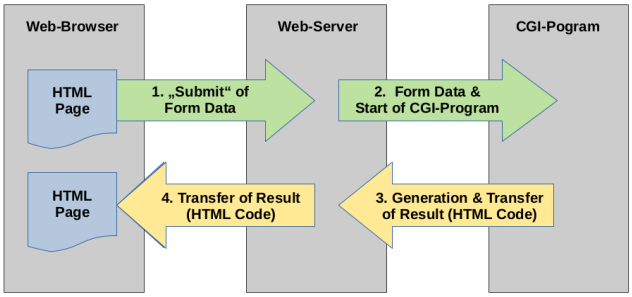

Abbildung 24.6.10.3.1: CGI-Protokoll

Der Client versendet eingegebene Formulardaten entweder mit der Get- oder Post-Methode an den HTTP-Server, wenn zum Beispiel eine Taste des Typs 'Submit' angeklickt wird. Die übertragenen Formulardaten 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 Formulardaten für den Zugriff durch das CGI-Programm bereit. Daraufhin wird das CGI-Programm gestartet, dass auf Umgebungsvariablen sowie Formulardaten zur weiteren Verarbeitung zugreifen kann. Das CGI-Programm erzeugt daraufhin als Resultat HTML-Code, der das Ergebnis in geeigneter Weise darstellt. Dieser HTML-Code 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 solche, mit denen die Generierung von mehrzeiligem HTML-Code einfach möglich ist. Die Spezifikation des Common Gateway Interfaces (CGI) kann unter dem folgenden Link nachgeschlagen werden: https://datatracker.ietf.org/doc/html/rfc3875.

Das folgende Beispiel soll die Verwendung der CGI-Schnittstelle demonstrieren. Dafür verwendet es eine HTML-Seite mit einem CGI-Aufruf und eine Gambas-Webpage-Applikation als CGI-Applikation. Die HTML-Seite enthält ein Formular, in das die beiden Operanden eingegeben werden und die Operation ausgewählt wird. Nach dem Abschicken der Formulardaten übernimmt die CGI-Applikation die erzeugten Umgebungsvariablen, um die einfachen Rechenoperationen durchzuführen sowie die Ergebnisse zu präsentieren.

Erzeugen Sie zunächst mit einem Texteditor Ihrer Wahl eine HTML-Datei mit dem u.a. Inhalt und speichern Sie die Datei unter dem HTML-Verzeichnis Ihres Servers unter dem Namen calc.html ab. In diesem Fall kommt das User-Verzeichnis zum Einsatz. Informationen zum angegebenen User-Verzeichnis finden Sie im Abschnitt '24.6.10.9.3 Modul UserDir'. Achten Sie auf die rot markierten Inhalte. Passen Sie den Usernamen und den Pfad zum CGI-Script entsprechend Ihrer Umgebung an:

HTML-Datei (calc.html) mit CGI-Aufruf:

```
<html>
<head>
   <Title>Calculator</Title>
   <meta charset="UTF-8">
</head>
<body style="font-family:Arial,Helvetica">
   <center>
     <h1>"Super Computer"</h1>
   </center>
   <p>
   <form method="post" action="http://localhost/~claus/cgi-bin/calc.gambas">
      <center>
      <p> <p>
      <input type="text" size="15" name="op1" value="0.0">
      <select name="op">
      <option value="1">+ (Add)
      <option value="2">- (Subtract)
      <option value="3">* (Multiply)
      <option value="4">/ (Divide)
      </select>
      <input type="text" size="15" name="op2" value="0.0">
      <p>
      <input type="reset" value="Reset Form">
       
      <input type="submit" value="Show Result">
      </center>
   </form>
</body>
</html>
```

Erzeugen Sie jetzt die CGI-Applikation, indem Sie in der Gambas-IDE ein neues Projekt vom Typ 'Web application' anlegen und in Main.webpage den folgenden Quelltext eingeben:

CGI-Applikation mit HTML-Ausgabe:

```
<%
Dim OP, OP1, OP2 As String
Dim sResult As String

'-- Get Request Parameters
OP1 = Request["op1"]
OP = Request["op"]
OP2 = Request["op2"]

'-- Calculate the result
Select Case OP
    Case 1
       sResult = Format(CFloat(OP1) + CFloat(OP2), "#,###,##0.######")
    Case 2
       sResult = Format(CFloat(OP1) - CFloat(OP2), "#,###,##0.######")
    Case 3
       sResult = Format(CFloat(OP1) * CFloat(OP2), "#,###,##0.######")
    Case 4
       sResult = Format(CFloat(OP1) / CFloat(OP2), "#,###,##0.######")
End Select
%>
<!-- Submit the result -->
<!DOCTYPE html>
<html lang="de">
  <head>
    <title>Result-HTML</title>
    <meta charset="utf-8">
    <style>
      body {background-color: #FFFFFF; font-family: Arial; font-size:10px; colo>
      h1 {text-align: left; font-family: Arial; font-size: 24px;}
    </style>
  </head>
  <body>
    <h1>The result is: <%= sResult%> </h1>
  </body>
</html>
```

Erzeugen Sie dann die ausführbare Datei "web_calc.gambas". Speichern Sie diese im CGI-Verzeichnis Ihrer Server-Umgebung ab; zum Beispiel unter /home/claus/public_html/cgi-bin. Um diesen "Web-Rechner" zu starten, öffnen Sie jetzt die HTML-Datei (direkt oder per URL über den HTTP-Server):

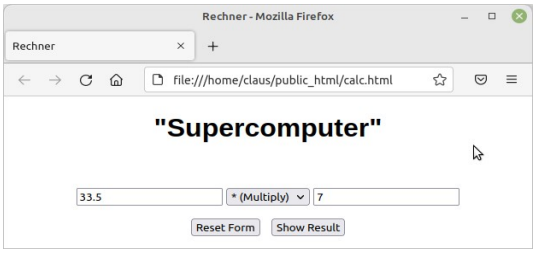

Abbildung 24.6.10.3.2: HTML-Formular

Zuerst geben Sie für die beiden Operanden reelle Zahlen ein. Dann wählen Sie die Rechenoperation aus. Abschließend klicken Sie auf 'Show Result'. Es sollte das Rechenergebnis in einem weiteren Webbrowser-Fenster anzeigt werden:

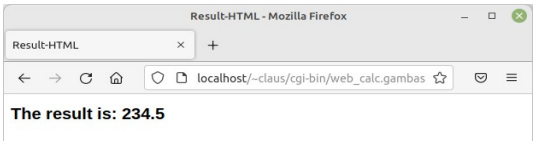

Abbildung 24.6.10.3.3: Ergebnisanzeige

Um das Ergebnis berechnen zu können, liest die Web-Applikation calc.gambas über Request.Post die drei Werte der 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 angezeigt.

24.6.10.4 Besonderheiten der verschiedenen Skriptsprachen für dynamische Webseiten

24.6.10.4.1 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 zwischen die Anfangs-Instruktion <?php und Abschluss-Instruktion ?> gesetzt. Anders als bei 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-whatis.php sehr hilfreiche Informationen in verschiedenen Sprachen zur Verfügung.

24.6.10.4.2 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 Webbrowser durch einen Interpreter abgearbeitet wird. Das erfordert das Vorhandensein einer nutzerseitigen JavaScript-Umgebung. Diese steht für alle gängigen Betriebssystem unter BSD-Lizenz zur Verfügung. JavaScript zeichnet sich durch eine C-ähnliche Syntax aus und ist objektorientiert. Seit HTML5 kann JavaScript-Code auch in der folgenden Form in HTML eingebettet werden:

```
<!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>
```

Auf der Webseite https://www.w3schools.com/html/tryit.asp?filename=tryhtml_basic_document können Sie den o.a. HTML-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

24.6.10.4.3 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 mit einem Print-Befehl (hier im <<Page;...Page-Block) 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
```

24.6.10.4.4 Python

Die Programmiersprache Python wird aufgrund ihrer starken Verbreitung auf den gängigen Betriebssystemen auch gern auf Servern als Skript-Sprache für das CGI-Protokoll verwendet. Die Programme werden durch einen Interpreter abgearbeitet. Es folgt ein einfaches 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 Skriptsprache kennenlernen möchte, kann sich auf der folgenden Webseite die notwendige Grundlagen erarbeiten: https://www.w3schools.com/python/default.asp .

24.6.10.5 Ausführen und Testen von Gambas Web-Applikationen

24.6.10.5.1 Der Embedded-Webserver

Zur Ausführung der Projektarten Web-Form-Applikation und Web-Applikation können Sie alternativ zu einem installierten Webserver als Webserver den sogenannten 'Embedded Server' verwenden, der mit der IDE zur Verfügung gestellt wird oder einen echten Webserver, den Sie sich nach einer weiter unten folgenden Beschreibung 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 Server' der IDE ist standardmäßig deaktiviert und muss zur Verwendung im Haupt-Menü der IDE unter 'Debuggen> Konfiguration> Debugger> 'Eingebetteten HTTP benutzen' aktiviert werden.

24.6.10.5.2 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 erzeugt, sie in das CGI-Verzeichnis des Servers kopiert, dann den Browser startet und die URL aufruft:

```
http://localhost/~claus/cgi-bin/wp_environment.gambas
```

Dieser Arbeitsablauf ist dank einer sehr nützlichen Funktion der IDE auch automatisierbar und wird in den nächsten Abschnitten beschrieben.

Lokaler Server mit einem CGI-Verzeichnis unter User-Rechten

Wenn Sie einen lokalen Test-Webserver zur Verfügung haben, 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/claus/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 zum Beispiel den Pfad ein:

```
/home/claus/public_html/cgi-bin/webapp.gambas
```

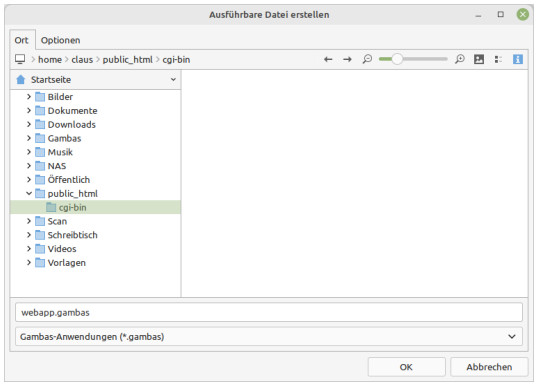

Abbildung 24.6.10.5.1: Verzeichnis-Auswahl

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:

```
Syntax:       firefox localhost/~<UserName>/cgi-bin/<GambasAppName>
Beispiel:     firefox localhost/~claus>/cgi-bin/wepapp.gambas
```

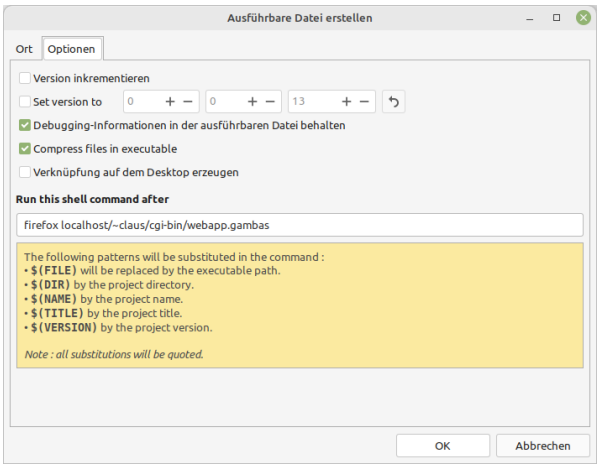

Abbildung 24.6.10.5.2: Auswahl 'Optionen' und Befehlseingabe

Die IDE speichert das eingegebene 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 Web-Applikation im Firefox-Browser automatisch aufgerufen.

Lokale Server 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 Erzeugung der ausführbaren Gambas-Datei ein automatischer Start des Webbrowsers und das 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 zum Beispiel

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

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.

Gnome/Cinnamon-Desktop:

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

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 die Webseite automatisch aufgerufen.

Debugging von Web-Applikationen

Das Debugging von Gambas-Web-Applikationen muss in vielen Bereichen anders vorgenommen werden als bei Desktop-Applikationen. Folgende Praxis hat sich 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-Browser mit weitreichenden Entwickler-Funktionen für verschiedene Zwecke.
- Verwendung des Embedded-Webservers, besonders bei der Programmierung von Gambas Web-Form-Applikationen, der ab Version 3.17.2 aktiviert werden muss.

- Verwendung eines echten Webservers, besonders bei der Projektart Gambas Web-Applikationen.
- Ausgabe des HTML-Codes in der IDE-Konsole, was jedoch die Deaktivierung des Embedded-Webservers erfordert.

24.6.10.6 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 es, 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 folgenden Regeln gelingt das für alle Formen von Web-Applikationen:

- Alle Dateien, die mit großer Wahrscheinlichkeit nicht ausgetauscht werden müssen wie Icons, werden wie üblich innerhalb des Projektordners (z.B. in Public oder dessen Unterverzeichnissen) gehalten und werden damit Bestandteil der kompilierten Gambas-Datei.
- Alle Dateien mit Daten, die austauschbar sein sollen, wie z.B. Datenbanken, werden innerhalb des Projektes im 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.

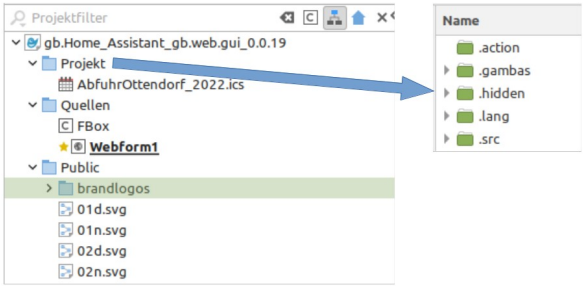

Abbildung 24.6.10.6.1: Verzeichnis-Struktur

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

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

Da auf einem Webserver mehrere Web-Applikationen zum Einsatz kommen können, empfiehlt es sich, dort die Dateien der verschiedenen Applikationen in getrennte Ordner zu kopieren, zum Beispiel:

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

oder 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, können Sie 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, ist unter 'Fernzugriff auf den Lighttpd-Server' beschrieben.

24.6.10.7 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 darstel-

len. Gambas Web-Applikationen können ebenfalls davon Gebrauch machen.

Bei der Verwendung in Web-Form-Applikationen ist es dazu erforderlich, dass man eine Datei mit dem Namen style.css in den Projektverzeichnis Public einfügt. Damit GUI-Elemente von Web-Form-Applikationen dieses StyleSheet 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 sollten Sie CSS-Dateien ebenfalls 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, um 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 einem Style Sheet 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 CSS genauer kennenlernen möchte, der kann sich beispielsweise auf der Webseite https://www.w3schools.com/css/ informieren.

24.6.10.8 Installation und Einrichtung eines Lighttpd Webservers

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.

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

Die installierte Version von Lighttpd können Sie so anzeigen:

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

Durch die Installation wird der Daemon auf Linux-Mint-Dstributionen automatisch gestartet. Das lässt sich so überprüfen:

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

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 Webserver 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 als 'Anderer' das Lese- und Ausführ-Recht gewährt werden.

Der Webserver kann bereits nach der Installation mit seiner Basiskonfiguration getestet werden. Dazu rufen Sie im Webbrowser

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

auf, weil der Webserver beim Aufruf automatisch nach einer 'index'-Seite im Webordner sucht. Der Webbrowser zeigt daraufhin eine vorinstallierte Webseite, die wichtige Informationen enthält:

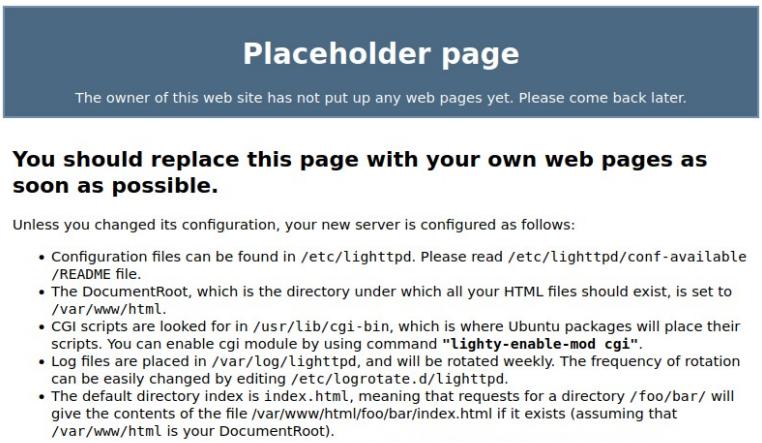

Abbildung 24.6.10.8.1: Informationsseite

24.6.10.8.2 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 folgendem Kommando erwiesen, weil neben dem Status auch auf mögliche Fehlerursachen hingewiesen wird:

```
$ service lighttpd status
```

24.6.10.8.3 Webserver-Steuerung

Die folgenden Aufrufe in einem Terminal steuern den Webserver Lighttpd mit den in der Liste aufge-

führten 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
```

24.6.10.8.4 Konfiguration des Webservers

Vor der Durchführung von Änderungen an der Konfiguration sollte die Original-Konfigurationsdatei des Server stets 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 nur die eine Zeile nach `# Insert changes here:` auf den angegebenen Text zu ändern.

Öffnen Sie die Konfigurationsdatei `lighttpd.conf`:

```
$ 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" )
# Insert changes here:
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 Konfigurationsdatei vorgenommen wurden, muss der Webserver veranlasst werden, die Konfiguration neu einzulesen:

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

24.6.10.9 Erweiterung des Webservers

Zur Erweiterung des Servers stehen eine Reihe von Modulen zur Verfügung; u.a. auch die folgenden zwei:

- 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

deren Einrichtung und Aktivierung in den nächsten Absätzen beschrieben wird.

Zur Deaktivierung von Modulen können Sie jederzeit das Kommando

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

verwenden. Danach müssen Sie den Server veranlassen, die geänderte Konfiguration zu laden:

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

24.6.10.9.1 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 passende 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

```
$ sudo lighttpd-enable-mod cgi
```

aktiviert und auch in diesem Fall muss der Server die veränderte Konfiguration mit dem folgenden Kommando neu einlesen:

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

Test mit einer Web-Form-Applikation

Testen können Sie die neue Funktionalität, indem Sie zum Beispiel eine dynamische Web-Seite anlegen. Erzeugen Sie zuerst im Home-Verzeichnis den Ordner ~/Gambas. Erzeugen Sie dann ein neues

Projekt vom Typ "Web form application". Nennen Sie das Projekt `Web_Form_App`. Speichern Sie es abschließend im Ordner ~/Gambas ab.

Das neue Projekt erzeugt Ihnen als Einstieg automatisch eine minimale Hallo-Welt-Applikation, welche neben Text und einem Button auch die Uhrzeit als dynamisches Element anzeigt und sofort einsatzbereit ist.

- Aktivieren Sie zuerst im Hauptmenü der IDE unter Debuggen>Konfiguration...Debugger die Option 'Eingebetteten HTTP-Server benutzen".
- Starten Sie in der IDE die Web-Applikation. Jetzt sollten Sie folgendes im Webbrowser sehen:

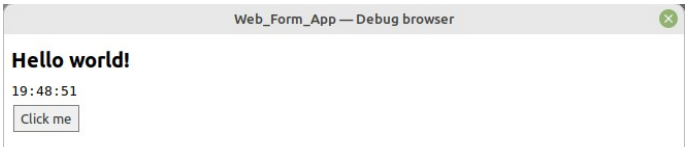

Abbildung 24.6.10.9.1: Ansicht Web-Form-Applikation (Debug-Modus)

Um diese Minimal-Web-Applikation auch auf dem Lighttpd-Server laufen zu lassen, erzeugen Sie zuerst die ausführbare Datei 'Web_Form_App.gambas'.

Kopieren Sie dann diese Datei mit Root-Rechten nach /usr/lib/cgi-bin:

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

Starten Sie abschließend den Webbrowser und geben Sie diese URL ein:

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

Das Ergebnis im Webbrowser sollte so aussehen:

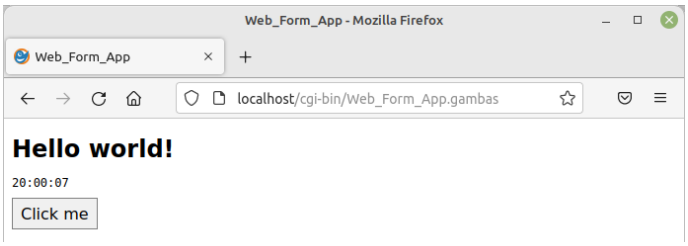

Abbildung 24.6.10.9.2: Ansicht Web-Form-Applikation

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 Webservers in unveränderter Form funktioniert. Das Zugriffsrecht als Rechte-String ist rwx-rx-rx oder in Worten:

- Besitzer: lesen, schreiben, ausführen.
- Gruppe: lesen, ausführen und
- Andere: lesen, ausführen.

wobei 'ausführen' für alle CGI-Skripte unverzichtbar ist.

Test mit einem Perl-Skript

Legen Sie eine Datei mit

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

an und fügen Sie folgenden Perl-Programm-Code 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 ein Skript als Text-Datei bei ihrer Erzeugung nicht mit den notwendigen oktalen Zugriffsrechten 755 ausgestattet. Deshalb müssen diese Rechte zum Beispiel im Zielverzeichnis angepasst werden – was schnell erledigt ist:

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

Im Webbrowser wird die Webseite so aufgerufen:

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

Das ist die Ansicht im Webbrowser:

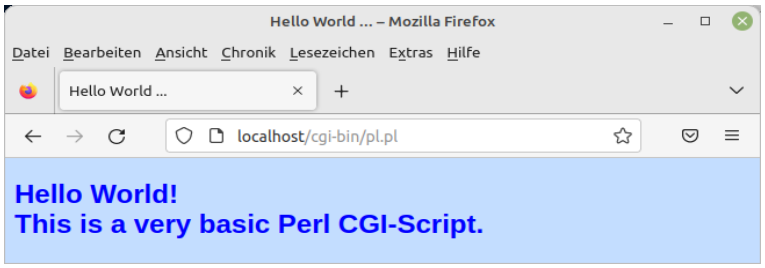

Abbildung 24.6.10.9.3: Anzeige im Webbrowser

Test mit einem Python-Skript

Legen Sie dazu eine Datei an

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

und fügen Sie folgenden Python-Programm-Code 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 erforderlichen Zugriffsrechte gesetzt werden, was mit

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

schnell getan ist. Im Webbrowser wird die Webseite so aufgerufen:

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

und das Ergebnis sehen Sie hier:

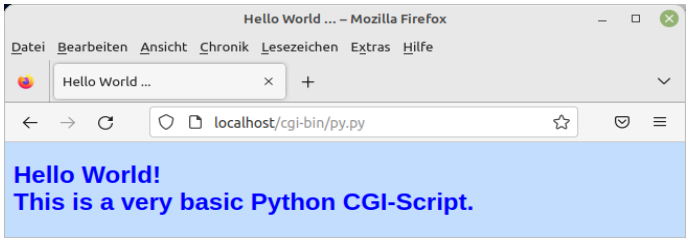

Abbildung 24.6.10.9.4: Anzeige im Webbrowser

Test für ein Gambas-Skript

Für diesen Test muss das Gambas-Paket gambas3-scripter installiert sein, was bei Bedarf so durchgeführt werden kann:

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

Legen Sie die Datei env.gbw3 in /usr/lib/cgi-bin/ an:

```
$ sudo nano /usr/lib/cgi-bin/env.gbw3
```

Fügen Sie dann den folgenden Gambas-Quelltext ein:

```
#!/usr/bin/env gbw3
<% DIM sEnv AS String
<!-- Variable declaration must come before any HTML -->
<!DOCTYPE html>
<html lang="de">
  <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] %> </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.gbw3
```

Im Webbrowser wird die Webseite aufgerufen

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

und die Anzeige sieht so aus:

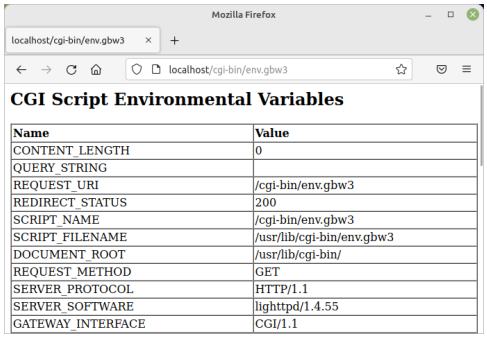

Abbildung 24.6.10.9.5: Anzeige der Umgebungsvariablen im Webbrowser

24.6.10.9.2 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 so sicherstellen können:

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

Die zugehörige Konfigurationsdatei unter /etc/lighttpd/conf-available/10-fastcgi.conf bleibt unverändert. Falls diese dennoch geändert werden soll, sichern Sie 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 einem Editor Ihrer Wahl

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

Danach sollten Sie die Datei /etc/php/7.4/cli/php.ini

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

öffnen und den Wert von cgi.fix_pathinfo auf 1 stellen oder das führende Semikolon in der zutreffenden Zeile entfernen – wenn das erforderlich ist.

Das Modul wird abschließend aktiviert:

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

Diese Aktivierung schließt das Modul fastcgi durch eine Abhängigkeit mit ein! Auch hier muss der Server die veränderte Konfiguration neu einlesen:

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

Um die neue Funktion des Servers zu testen, erzeugen Sie die Skript-Datei php.php

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

und fügen folgenden PHP-Quelltext ein:

```
<?php
echo "<!DOCTYPE html>";
echo "<html lang=\"en\">";
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 so aufgerufen:

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

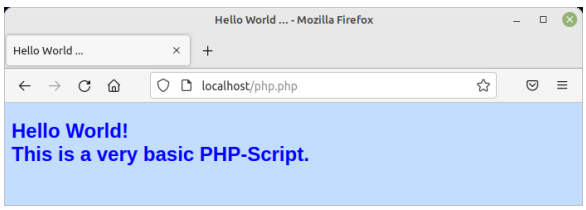

Abbildung 24.6.10.9.6: Anzeige im Webbrowser

24.6.10.9.3 Modul UserDir

Da man auf die bisher verwendeten Server-Verzeichnisse /var/www und /usr/lib/cgi-bin nur mit Root-Rechten zugreifen kann, wäre es eine erhebliche Vereinfachung, wenn man die Webseiten im eigenen Home-Verzeichnis ablegen könnte. Genau dafür 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 folgendem Befehl die Konfiguration neu laden:

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

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 der Installation des Betriebssystems auf 750 gesetzt, was eine Verwendung des Userdirs verhindert! Mit dem folgenden Befehl ist das jedoch schnell in Ordnung gebracht:

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

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 anschließend 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; colo>
      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 die Webseite im Webbrowser in leicht veränderter Weise auf:

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

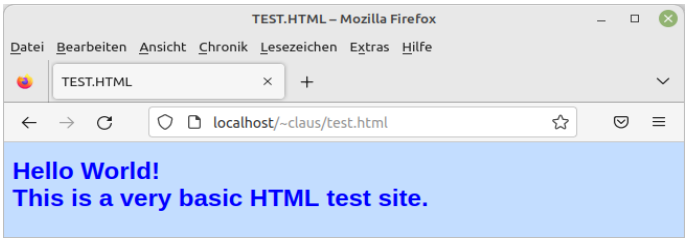

Abbildung 24.6.10.9.7: Anzeige im Webbrowser

Test mit PHP-Skripten

PHP-Skripte können ebenfalls in dem Verzeichnis ~/public_html abgelegt werden. Da bereits das PHP-Skript ph.php im Standard-Verzeichnis /var/www/html angelegt wurde, kann das für Testzwecke

```
$ 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 Webbrowser aufgerufen und angezeigt werden:

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

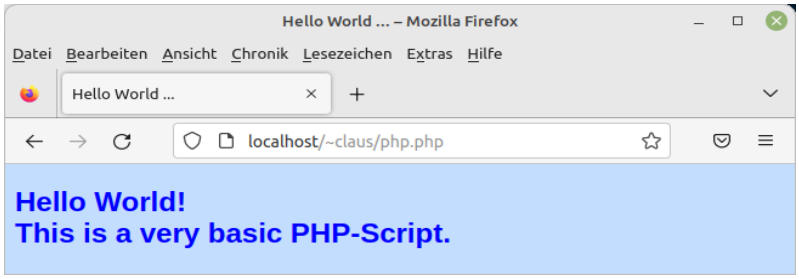

Abbildung 24.6.10.9.8: Anzeige im Webbrowser

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 dafür das Unterverzeichnis `cgi-bin` an:

```
$ mkdir /home/$USER/public_html/cgi-bin
```

Legen Sie CGI-Web-Applikationen, die Sie bisher unter /usr/lib/cgi-bin abgelegt haben, jetzt im Verzeichnis ~/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 die Dateien über das Zugriffsrecht 775 (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 sichern Sie zunächst die Originaldatei

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

und ändern dann die Datei 10-cgi.conf mit einem Editor Ihrer Wahl:

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

Das ist der Inhalt der Konfigurationsdatei conf-available/10-cgi.conf:

```
# /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"

# Enter the changes here:
# Begin
   cgi.assign = ( ".pl"  => "/usr/bin/perl",
```

```
                  ".cgi" => "/usr/bin/perl",
                  ".py"  => "/usr/bin/python3",
                  ".gbw3" => "/usr/bin/gbw3",
                  ".gambas" => "/usr/bin/gbr3" )
# End

## 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 Lighttpd die geänderte Konfiguration neu einlesen:

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

und rufen das Perl-Skript auf

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

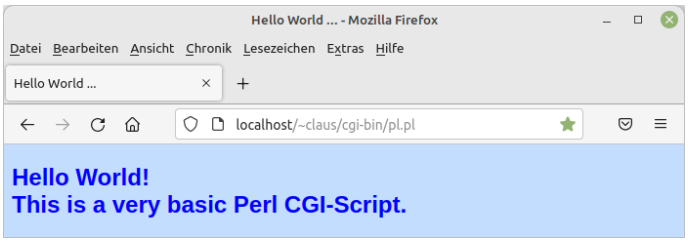

Abbildung 24.6.10.9.9: Anzeige im Webbrowser

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.

24.6.10.9.4 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 Datei /etc/lighttpd/conf-available/10-ssi.conf konfigurieren:

```
$ sudo nano /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 sowie Texte aus zwei externen Text-Dateien einliest und in den HTML-Quelltext der Webseite einbindet, werden die SHTML-Datei ssitest.shtml sowie die beiden Text-Dateien im Downloadbereich bereit gestellt.

Abbildung 24.6.10.9.10: 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 erforderlichen Rechte zu setzen:

```
$ sudo chmod 755 $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>Dynamic HTML with Server Side Includes (SSI)</h1>
    <p>Date and current time on the server:
    <!--#config timefmt="%d.%m.%Y, %H:%M" --><!--#echo var="DATE_LOCAL" --> Uhr<br/></p>
    Installed 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-Protocol: <!--#echo var="SERVER_PROTOCOL" --><br />
    Request-Method: <!--#echo var="REQUEST_METHOD" --><br />
    Document root directory: <!--#echo var="DOCUMENT_ROOT" --><br />
    Request-URI: <!--#echo var="DOCUMENT_URI" --><br />
    File path: <!--#echo var="SCRIPT_FILENAME" --><br />
    Name of the sHTML file: <!--#echo var="DOCUMENT_NAME" --><br />
    File modified on <!--#config timefmt="%d.%m.%Y um %H:%M" --><!--#echo var="LAST_MODIFIED" --> Uhr<br />
    Client-Webbrowser: <!--#echo var="HTTP_USER_AGENT" --><br />
    <br />
    Can you make heads or tails of the following text - inserted from an external file?
    <p><!--#include file="texte/loremipsum.html" --></p>
    Top secret? Ah - yes. That had been suspected for a long time!<br /><br />
    Source code of the sHTML file: <!--#echo var="DOCUMENT_NAME" -->
    <hr />
    <p><!--#include file="texte/ssitest.txt" --></p>
  </body>
</html>
```

So rufen Sie die 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 deutlich.

24.6.10.10 Fernzugriff auf den Webserver Lighttpd

Wenn Sie den Webserver Lighttpd in ihrem lokalen Netzwerk auf einer separaten Hardware wie Computer oder NAS (entfernter Server) installiert haben, dann bietet sich besonders das SSH-Protokoll an, um Web-Seiten/Dateien einfach und sicher auf den entfernten 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 in der Regel auf jeder Linux-Plattform, also auch auf Ihrem Entwicklungssystem, vorinstalliert. Falls noch kein SSH-Server auf dem entfernten Server vorhanden ist, so bietet sich das OpenSSH-Server-Paket an, das dort schnell installiert ist:

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

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 '192.168.123.100 (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:DIb+yYTRkuzurn7+qjJcNIOZOjI2tveaazDqrDA38YE.
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 entfernten Server angemeldet. Die Verbindung kann auch wieder mit `exit` aufgehoben werden:

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

Da sich in einem lokalen Netztwerk meist ein DHCP-Server befindet, können Sie die Eingabe der IP-Adresse umgehen, indem Sie den (Host-)Namen dee Servers angeben wie im folgenden Fall:

```
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+qjJcNIOZOjI2tveaazDqrDA38YE.
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 abermals `exit` ein. Danach befinden 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.

24.6.10.10.1 Übertragung von Webseiten auf den Server

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

- Datei-Manager oder SSH-Kommandos in einer Konsole

Die modernen Datei-Manager von Linux-Distributionen beherrschen das SSH-Protokoll und Sie können sich auf sehr einfache Weise damit bei einem SSH-Server anmelden. Hier ein beispielhafter Dialog des Datei-Managers Nemo (Datei>Mit Server verbinden):

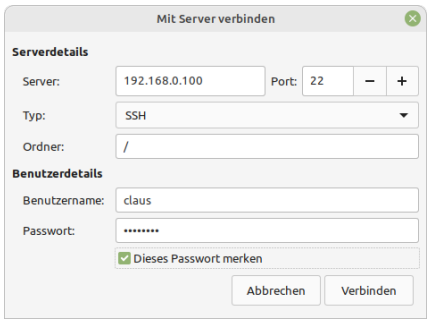

Abbildung 24.6.10.10.1: Dialog-Anzeige im Datei-Manager

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 SSH-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 mit Inhalt rekursiv auf den Server übertragen wollen, dann nutzen Sie folgende Syntax:

```
$ scp -r <Quell-Verzeichnis> <SSH-Username>@<SSH-Server-IP>:<Ziel-Verzeichnis>
```

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 im Gambas-Buch im Detail beschrieben ist:

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

Beim Einsatz eines Webservers auf einer separaten Hardware im lokalen Netzwerk, auf den auch andere Netzwerk-Clients 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/UserName/cgi-bin/pl.pl). Damit der Aufruf von Webseiten ohne Angabe eines User-Namens erfolgen kann (zum Beispiel mit http://192.168.1.100/cgi-bin/pl.pl) müssen CGI-Applikationen in das Verzeichnis

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

und HTML-Seiten in das Verzeichnis

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

des Servers kopiert werden, was bei bestehender SSH-Verbindung für einzelne Dateien - beispielsweise für ein Perl-CGI-Script - so vorgenommen werden kann:

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

Der Besitzer wird dann noch auf "root" gesetzt:

```
$ 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
```

24.6.10.11 Installation von Datenbank-Systemen für Web-Applikationen

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

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 so vorgenommen werden:

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

Datenbanksysteme mit Client-Server-Architektur

Für Herstellung eines Datenbanksystems mit Client-Server-Umgebung sind neben der Installation auch geeignete Konfigurationen vorzunehmen. Diese Quelle bietet umfangreiche Informationen über die Einrichtung und Verwendung solcher Datenbanksysteme:

```
https://www.gambas-buch.de/doku.php?id=k22:start
```

24.6.10.12 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.

Bei der 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 die Gambas-IDE mit zu installieren, womit sich der Aufwand zur Übertragung einer Webseite auf den Server minimiert. Ein Webbrowser sollte für diesen Fall auch verfügbar sein.

Zur Installation 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)