Tutorium

CGI

Stefan Probst
Kirchenplatz 14/6
4209 Engerwitzdorf
stefan.probst@spps-consulting.de

CGI-Programmierung

Dieses Tutorium erklärt anhand eines praktischen Beispieles die Programmierung des Common Gateway Interface (CGI). Zudem werden HTML-Formulare vorgestellt, die das Haupteinsatzgebiet von CGI sind.

CGI ist ein beliebtes und einfaches Mittel um dynamische Webseiten zu realisieren. Dynamische Webseiten liegen nicht in Form von HTML-Dateien am Server sondern werden zur Laufzeit vom Server auf Anfrage erstellt. Somit kann der Benutzer aktiv in die Kommunikation mit dem Webserver einbezogen werden, da dieser auf Eingaben und Anfragen des Benutzers gezielt reagieren kann und die Ausgabe entsprechend gestalten kann.

Dieses Tutorium ist als begleitendes Dokument zur Lehrveranstaltung gedacht. Sie finden darin die Folien, die in der Lehrveranstaltung verwendet werden, und weiterführende Informationen sowie Programmbeispiele, die Ihnen die CGI-Programmierung sowie HTML-Formulare näher bringen sollen.

SYMBOL-LEGENDE	
ⓘ	Informationen
!	Wichtige Hinweise
🖉	Beispiele
?	Verständnisfragen

Verwendung des Skripts: In diesem Skriptum werden Sie immer wieder auf verschiedene Symbole und Schreibweisen stoßen.

Spezielle Zeichen werden durch einfache Anführungszeichen (‚) gekennzeichnet. Kommandos werden innerhalb doppelter Anführungszeichen („“) erwähnt. Variable Teile innerhalb eines Kommandos sind *kursiv* gedruckt, genauso wie betonte Wörter. Am Seitenrand werden Sie immer wieder Symbole finden, die auf spezielle Informationen hindeuten.

Lehrinhalte und -ziele

Dieses Tutorium beschäftigt sich primär mit der CGI-Programmierung und stellt Ihnen die Programmbibliothek *cgihtml* vor, die eine einfache Realisierung von CGI Programmen mittels der Programmiersprache *C* erlaubt.

Da CGI häufig für die Auswertung von HTML-Formularen genutzt wird, werden HTML-Formulare und deren Realisierung näher vorgestellt.

Als zugrunde liegende Plattform wird der Apache Web-Server benutzt.

Konfiguration des Apache Web-Servers

Apache Web-Server

- Häufig genutzter Web-Server (http://httpd.apache.org)
- Installation
  - `./configure --prefix` *`Zielverzeichnis`*
  - `make`
  - `make install`
- Konfiguration
  - `Zielverzeichnis/conf/httpd.conf`
  - Einstellen des Ports: `LISTEN` *`Portnummer`*
  - CGI-Programmverzeichnis: `<DIRECTORY>`-Sektion

Das Apache HTTP Server Projekt hat sich zum Ziel gesetzt, einen modernen Webserver für gängige Betriebssysteme wie UNIX und Windows bereitzustellen. Ziel des Projektes ist einen sicheren, effizienten Webserver bereitzustellen, der konform zu aktuellen Standards des http Protokolls ist.

Der Webserver Apache ist ein Open-Source Projekt und kann somit kostenlos inklusive Quellcode erworben werden. Zudem ist der Webserver für alle heute gängigen, aber auch exotischere Betriebssysteme verfügbar. Eine im Oktober 2003 durchgeführte Studie ergab, dass Apache der meist genutzte Webserver (mehr als 64%) im Internet ist. Mehr Informationen zum Apache Webserver können unter der Internetadresse *http://httpd.apache.org* gefunden werden.

Apache Installation

In weiterer Folge wird die Installation von Apache 2.0 unter IBM's AIX beschrieben. Die Installation auf anderen Betriebssystemen sieht ähnlich aus.

1.) Entpacken der Installationsdatei: „gunzip httpd-2*.gz".

2.) Extrahieren der Dateien aus dem Archiv: „tar –xvf httpd-2*.tar". Diese Operation erzeugt Ihnen ein neues Unterverzeichnis mit dem Namen der Archivdatei (ohne die Endung „.tar'). In diesem Verzeichnis befindet sich nun der Quellcode von Apache. Dieser muss zuerst auf der entsprechenden Zielmaschine übersetzt werden. *ACHTUNG: Die Archivdatei wird beim extrahieren nicht gelöscht.* Um Platz zu sparen, sollten Sie diese manuell mit dem Kommando „rm httpd-2*.tar" löschen.

3.) Übersetzung und Installation von Apache: Wechseln Sie nun zuerst in das Verzeichnis, in dem sich die Apache Quelldateien befinden. Die Datei

‚INSTALL' gibt nähere Informationen darüber, wie Apache übersetzt und installiert werden muss. In UNIX müssen folgende Schritte zur Installation von Apache ausgeführt werden:

a. Festlegen des Installationsverzeichnisses mit dem Befehl „./configure --prefix *Zielverzeichnis*". Geben Sie bei Zielverzeichnis ein Verzeichnis an, in dem Sie Lese- und Schreibrechte haben. Die Installationsroutine prüft gleichzeitig das System und das Vorhandensein notwendiger Softwarepakete.

b. Starten Sie den Übersetzungsvorgang durch die Eingabe des Befehls „make".

c. Die Installation erfolgt durch die Eingabe des Befehls „make install".

4.) Der Apache Webserver ist nun installiert und kann mit dem Befehl „*Installastionsverzeichnis*/bin/apachectl start" gestartet werden. Allerdings empfiehlt es sich, den Webserver vorher entsprechend zu konfigurieren.

Apache Konfiguration

Die Konfiguration des Apache Webservers erfolgt in der Datei ‚httpd.conf' die sich im Unterverzeichnis ‚conf' des Installationsverzeichnisses befindet. Um den Webserver zu konfigurieren, muss diese Datei in einem beliebigen Texteditor geöffnet werden. In weiterer Folge werden nur die Konfigurationsdaten besprochen, die für dieses Tutorium relevant sind.

- Einstellen des Ports: Wird nicht mit dem Superuserkonto (‚root') gearbeitet sondern mit einem normalen Benutzerkonto (z.B. im Übungs-/Testbetrieb) so kann der Apache Webserver nicht an den Standardport 80 binden. In UNIX darf nur der Superuser an die so genannten *well-known* Ports (Ports bis 1024) binden. Um Apache an einen Benutzerport (höher 1024) zu binden, muss in der Zeile „LISTEN 80" das Port 80 gegen eine entsprechende andere Portnummer ausgetauscht werden.

- Einstellen des CGI Programmverzeichnisses: Der Webserver erwartet CGI Programme in einem speziellen Verzeichnis. Dieses Verzeichnis muss zuvor spezifiziert werden. Dazu muss in der Sektion <Directory> ein entsprechendes Verzeichnis eingegeben werden, in welchem der Webserver nach solchen Programmen sucht. In weiterer Folge ist diese Sektion als Beispiel abgebildet.

```
# CGI directory exists, if you have that configured.
#
<Directory "absoluter_Pfad_zum_CGI_Verzeichnis">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

Common Gateway Interface (CGI)

Common Gateway Interface (CGI)

- Ermöglicht den Aufruf von Programmen oder Skripts aus HTML-Seiten
  - Generierung von kontextbezogenen Seiten
  - CGI-Programme am Server ausgeführt
  - Client erhält generierte HTML-Seite
- Funktionsweise von CGI-Programmen
  - Lesen von Standardeingabe
  - Schreiben auf Standardausgabe
  - Webserver leitet Ein- und Ausgabe um

Die CGI-Schnittstelle

Die Common Gateway Interface (CGI) Schnittstelle ist eine Möglichkeit, Programme oder Skripts aus HTML-Seiten aus aufzurufen um selbst neue, kontextbezogene HTML-Seiten zu generieren. CGI Programme werden am Webserver selbst ausgeführt, verarbeiten auch dort die Daten und schicken lediglich HTML-Code an den Client zurück. Somit sind CGIs unabhängig vom verwendeten Browser auf der Clientseite.

ⓘ Es gibt keine Vorschriften dafür, in welcher Programmiersprache ein CGI-Programm geschrieben ist. Damit das Programm auf dem Server-Rechner ausführbar ist, muss es entweder für die Betriebssystem-Umgebung des Server-Rechners als ausführbares Programm kompiliert worden sein, oder es muss auf dem Server ein Laufzeit-Interpreter vorhanden sein, der das Programm ausführt. Wenn der Server zum Beispiel ein Linux-Rechner ist, führt er C-Programme aus, die mit einem Linux-C-Compiler zu einer ausführbaren Datei kompiliert wurden. Wenn der Server ein Windows-Rechner ist, können CGI-Scripts auch EXE-Dateien sein, die mit Compilern für C, Pascal, Visual Basic usw. erzeugt wurden. Die meisten heutigen CGI-Programme sind jedoch keine kompilierten Programme, sondern lediglich Scripts, die von einem Interpreter beim Aufruf ausgeführt werden. Der bekannteste und beliebteste Interpreter ist dabei der Perl-Interpreter.

Typische Anwendungen von CGI

CGI wird üblicherweise dann angewandt, wenn eine HTML-Seite Ressourcen des Webservers benötigt. Ein typischer Anwendungsfall ist der Zugriff auf eine Datenbank, die am Server liegt.

CGI wird aber auch angewandt, wenn Seiten dynamisch generiert werden sollen, allerdings keine Abhängigkeit zum verwendeten Client-Browser bestehen soll. Zudem erlaubt CGI eine Interaktion mit dem Benutzer und das gezielte Reagieren von Webseiten auf Eingaben des Benutzers.

Funktionsweise von CGI

CGI Programme verstehen sich als Datenparser und unterscheiden sich von der Programmierung nur kaum von herkömmlichen Programmen. Typischerweise liest ein CGI Programm Daten von der *Standardeingabe* und schreibt Informationen auf die *Standardausgabe*. Wird ein solches Programm allerdings von einem Webserver aufgerufen, erfolgt eine Ein- und Ausgabeumleitung. So leitet der Webserver die Eingabe so um, dass diese mit den CGI-Daten des Webservers gefüttert wird. Die Ausgabe wird wiederum zum Webserver umgeleitet, wobei der Webserver die Ausgabe als HTML-Datei interpretiert. Logischerweise muss die Ausgabe also HTML-Daten generieren. Programmtechnisch werden die HTML-Daten allerdings an die Standardausgabe geschrieben (mit *printf()*) und Eingaben vom Server von der Standardeingabe gelesen (z.B. *scanf()*).

Installation der CGI Bibliothek ‚cgihtml'

Funktionsbibliothek ‚cgihtml'

- Sammlung von C-Routinen
  - Parsen der übergebenen Daten vom Server
  - Dynamische Generierung von HTML-Seiten
- Einsatzgebiet
  - Auswertung von HTML-Formularen
  - Eingegebene Daten werden vom Webserver an Programm übergeben
  - Programm wertet mit Hilfe von cgihtml-Funktionen die Parameter aus.
- http://www.eekim.com/software/cgihtml

Die Programmbibliothek ‚cgihtml' ist eine Sammlung von Routinen, geschrieben in der Programmiersprache C, um WWW-Daten mittels CGIs zu parsen und ist in der Lage dynamisch HTML-Seiten zu generieren. ‚cgihtml' vereinfacht dabei die Aufgabe des Parsens von CGI-Eingaben und der Generierung von dynamischen HTML-Code.

Installation von cgihtml

Die Installation der Programmbibliothek erfolgt analog zur Installation des Webservers. Zuerst muss ‚cgihtml' herunter geladen werden (Quelle: http://www.eekim.com/software/cgihtml/). Die Bibliothek liegt wieder im Quelltext vor. Nach dem dekomprimieren (‚gunzip cgihtml*') und extrahieren (‚tar –xvf cgihtml*') muss die Datei mit dem Kommando „make" im entsprechenden Installationsverzeichnis übersetzt werden.

Das Kommando „make examples" erzeugt Beispielprogramme, welche die Benutzung von ‚cgihtml' zeigen und dessen Arbeitsweise veranschaulichen.

CGI zur Auswertung von HTML-Formularen

CGI wird hauptsächlich zur Verarbeitung von HTML-Formularen eingesetzt. Um den sinnvollen Einsatz von CGI und der Anwendung von ‚cgihtml' zu zeigen, wird deshalb zunächst kurz auf HTML-Formulare eingegangen und deren Anwendung gezeigt.

Installation des SSW CGI Dev-Environments

SSW CGI Dev-Environment

- Sammlung von Shell-Skripts zum einfachen generieren von CGI Programmen
  - `buildcgi` *`quelldatei(en)`* `[-o` *`zieldatei`*`]`
    - Übersetzt CGI-Programm (automatisches Einbinden von ‚cgihtml')
  - `installcgi` *`cgiprogramm`*
    - Installiert übersetztes CGI-Programm in Apache

Das SSW CGI Dev-Environment richtet eine strukturierte Umgebung für die Entwicklung von CGI-Programmen unter Verwendung des Apache Webservers ein. Dazu wird einerseits eine Verzeichnisstruktur angelegt, andererseits werden Kommandos zur einfacheren Entwicklung von CGI-Programmen angeboten.

Installation des CGI Dev-Enrivonments

Das CGI Dev-Environment setzt voraus, dass auf dem System der Apache Webserver und die Programmbibliothek ‚cgihtml' installiert sind. Dies wird zusätzlich von der Installationsroutine überprüft.

Die Installation erfolgt durch den Start des Shell-Skripts ‚makeCGIenv'. Das Skript erwartet einen Parameter, welcher das Zielverzeichnis bestimmt, in dem das Dev-Environment installiert werden soll.

Das Dev-Environmentverzeichnis besteht aus zwei Unterverzeichnissen:

- bin: Hier befinden sich zwei Skripts die ein einfaches Übersetzen und eine einfache Installation der CGI-Programme ermöglichen.
  - „buildcgi *quelldatei(en)* [–o *zieldatei*]": Übersetzt ein CGI-Programm.
  - „installcgi *cgiprogramm*": Installiert das CGI-Programm in Apache.
- source: Dieses Verzeichnis ist für die Entwicklung von CGI-Programmen vorgesehen. Der Quellcode von CGI-Programmen sollte hier gespeichert werden (nicht zwingend erforderlich).

HTML-Formulare

HTML-Formulare

- Möglichkeit der Eingabe von Daten durch Benutzer
- Stellen verschiedene Kontrollelemente zur Verfügung
  - Eingabe von Text, Auswahl von Optionen
- Eingegebene Daten werden an Aktionsziel (Mail oder Programm) zugestellt
  - Daten können von Programmen weiterverarbeitet werden

Allgemeines zu HTML-Formularen

HTML stellt die Möglichkeit zur Verfügung, Formulare bereitzustellen. Formulare werden typischerweise vom Benutzer ausgefüllt und stellen somit eine Möglichkeit dar, dass der Benutzer mit der Webseite interagieren kann. Formulare können somit hergenommen werden, um Daten vom Benutzer zu erfragen oder aber zur Steuerung der Seite durch den Benutzer. Dazu stellen HTML-Formulare unterschiedliche Kontrollelemente zur Verfügung, die eine unterschiedliche Dateneingabe ermöglichen.

Beim Erstellen eines Formulars wird festgelegt, was mit den Daten passieren soll. So können diese Daten beispielsweise per Mail an eine bestimmte Adresse versendet werden, oder aber von einem CGI-Programm am Server weiterverarbeitet werden.

Aufgaben von HTML-Formularen

Formulare können für unterschiedliche Aufgaben genutzt werden, wie z.B. das

- Einholen von Auskünften und Daten von Anwendern.
- Suchen in Datenbeständen (Eingabe des Suchbegriffes).
- Eingeben von Daten (z.B. Bestellliste, Online-Shop, Informationen).
- Steuern der Navigation der Seite.

Erstellen eines HTML-Formulars

In weiterer Folge wird gezeigt, wie HTML-Formulare in eigenen Webseiten erstellt werden können. Allerdings werden nur die wichtigsten Konzepte gezeigt. Für eine detaillierte Beschreibung der Möglichkeiten sei auf die Seite ‚selfhtml' (http://selfhtml.teamone.de) verwiesen.

Jedes HTML-Formular besteht aus einem Formularbereich, der durch den Bezeichner ‚<FORM>' eingeleitet wird und durch den Endbezeichner ‚</FORM>' abgeschlossen wird. Bei der Definition des Formularbereichs wird auch gleich angegeben, was mit den eingegebenen Daten zu geschehen hat.

```
<FORM ACTION="Daten_Aktion" METHOD="Übermittlungsmethode">
   Formularelemente
</FORM>
```

Das Attribut ‚ACTION' gibt dabei an, wie die Daten verarbeitet werden sollen. So kann hier entweder eine E-Mail Adresse mit vorangestelltem *mailto:* angegeben werden, um die Daten per Mail an die angegebene Adresse zu versenden (ACTION="mailto:name@nowhere.com"), oder aber eine Referenz zu einem ausführbaren Programm (CGI-Programm), welches die Daten am Server weiterverarbeitet (ACTION="http://www.domain.at/cgi-bin/sample.cgi").

Das Attribut ‚METHOD' spezifiziert, nach welcher HTTP-Übertragungsmethode die Daten an das Aktionsziel zugestellt werden. Hier kann zwischen folgenden Möglichkeiten ausgewählt werden:

- method="get": Hier werden die Daten als Parameter an die Aufrufadresse angehängt. Dazu wird an die normale Zieladresse ein ‚?' angehängt, danach folgenden die einzelnen Daten in der Form ‚*Parametername*=*Wert*'. Die einzelnen Parameter werden durch ein ‚&' verbunden. (.../sample.cgi?User=Stefan&LVA=NET4). Das verarbeitende Programm muss diese übergebene Zeichenkette auslesen und für die Datenverarbeitung parsen.

- Method="post": Hier werden die Daten des Formulars über Eingabeumleitung zur Verfügung gestellt. Das verarbeitende Programm muss die ankommenden Daten wie Benutzereingaben behandeln. Standardmäßig wird bei CGI diese Methode gewählt.

Innerhalb des Formularbereichs kann sich ganz normaler Text, optional angereichert mit HTML-Formatierungsbezeichner, befinden. Für die Eingabe von Daten sind allerdings spezielle Kontrollelemente von Nöten, die über eigene Bezeichnernamen verfügen.

Kontrollelemente zur Dateneingabe

Je nach gewünschter Art der Dateneingabe stehen verschiedene Kontrollelemente zur Verfügung.

Jedes Kontrollelement verfügt über einen Namen, der innerhalb des Formulars eindeutig sein muss. Je nach Art des Kontrollelements ist oft auch die Angabe eines Wertes oder Gruppierungsnamens nötig.

Textbox

Die Textbox eignet sich gut für einfache, kurze Texteingaben wie z.B. Namen, Suchbegriffe oder E-Mail Adresse.

Abbildung 1: Textbox

Die Bezeichnersyntax für eine Textbox lautet wie folgt:

```
<INPUT TYPE="text" NAME="name" SIZE="anzeige_größe"/>
```

Textbereichsbox

Die Textbereichsbox ist eine mehrzeilige Eingabemöglichkeit für Text, wie z.B. Adresse oder Nachrichtentext.

Abbildung 2: Textbereichsbox

Die Bezeichnersyntax für eine Textbereichsbox lautet wie folgt:

```
<TEXTAREA NAME="name" ROWS=anzahl_zeilen COLS=anzahl_spalten>
</TEXTAREA>
```

Die Textbereichsbox muss explizit durch den Endbezeichner </TEXTAREA> abgeschlossen werden. Jeglicher Text, der sich zwischen dem Anfangsbezeichner und Endbezeichner befindet, wird in die Textbox eingetragen (z.B. Textvorgaben).

Optionsschalter

Optionsschalter (*Radio Buttons*) sind eine Möglichkeit, einen Wert aus mehreren möglichen Optionen zu wählen. Es kann allerdings innerhalb einer Optionsgruppe immer nur *ein einziger* Wert gewählt werden.

Abbildung 3: Optionsschalter

Die Bezeichnersyntax für einen Optionsschalter lautet wie folgt:

```
<INPUT TYPE="radio" NAME="gruppe" VALUE="wert"/> Beschriftung
```

Optionen werden anhand ihres Namens gruppiert. Innerhalb einer Gruppe kann immer nur eine Option ausgewählt werden. Jede Option kann mit einem Wert versehen werden, der bei einer Auswahl dann an das Ziel als Wert des Gruppennamens geliefert wird.

Auswahlschalter

Auswahlschalter (*Checkboxen*) verhalten sich ähnlich wie Optionsschalter, erlauben aber eine mehrfache Auswahl von möglichen Optionen.

Abbildung 4: Auswahlschalter

Die Bezeichnersyntax für einen Auswahlschalter lautet wie folgt:

```
<INPUT TYPE="checkbox" NAME="gruppe" VALUE="wert"/> Beschriftg.
```

Auch Auswahlschalter werden gruppiert. Die Gruppierung erfolgt anhand ihres Namens. Je nach ausgewählter Option können pro Gruppe 0 oder mehrere Werte zurückgeliefert werden.

Auswahlliste

Die Auswahlliste (*Popup Box*) erlaubt das Auswählen einer Option aus einer Liste von möglichen Optionen.

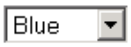

Abbildung 5: Auswahlliste

Die Bezeichnersyntax für eine Auswahlliste lautet wie folgt:

```
<SELECT NAME="name" SIZE="angezeigte_elemente">
  <OPTION SELECTED>Standardmäßig_gewähltes_Element</OPTION>
  <OPTION>Option</OPTION>
  <OPTION>...</OPTION>
</SELECT>
```

Das Attribut ‚SIZE' ist standardmäßig 1. Wird hier ein Wert größer 1 eingegeben, so wird die Auswahlliste als mehrzeilige Liste dargestellt. Die einzelnen Listenelemente werden als Optionen (‚OPTION') angegeben, wobei die standardmäßig ausgewählte mit ‚OPTION SELECTED' angegeben werden kann.

Aktionsschaltflächen

Die Aktionsschaltflächen starten bestimmte Verarbeitungsaktionen der Formulardaten. Es gibt zwei verschiedene Aktionen, nämlich ‚*submit*' und ‚*reset*'.

Abbildung 6: Aktionsschaltflächen

Formulardaten an Ziel senden (submit):

```
<INPUT TYPE="submit" VALUE="beschriftung"/>
```

Formulardaten zurücksetzen (reset):

```
<INPUT TYPE="reset" VALUE="beschriftung"/>
```

Erstes CGI-Programm

Erstes CGI-Programm

- Besteht aus HTML-Formular
  - <FORM>-Sektion
  - ACTION: Starten des CGI-Programms
  - METHOD: `post`
  - Mindestens Aktionsschaltfläche „Submit“
- CGI-Programm
  - Generierung einer HTML-Seite
  - ‚cgihtml‘-Funktionen
    - `html_header(), html_begin(), html_end()`

Die CGI-Programmierung wird nun anhand eines einfachen Beispiels gezeigt. Dazu wird am Webserver ein einfaches Formular angelegt, dass nur aus einer Aktionsschaltfläche besteht, die allerdings ein CGI-Programm startet.

HTML-Formular

Im ersten Schritt erstellen wir nun ein HTML-Formular, dass nur aus einer Aktionsschaltfläche besteht. Apache erwartet HTML-Seiten im Unterverzeichnis ‚htdocs’ des Installationsverzeichnisses. Dieses Verzeichnis bildet die Ausgangsbasis für das Dateisystem, wie es im Webbrowser angezeigt werden kann. Für unser erstes Beispiel sollten wir mit dem Kommando „mkdir firstSample“ ein neues Unterverzeichnis ‚firstSample’ im htdocs-Verzeichnis anlegen. In diesem neuen Verzeichnis erstellen wir nun in einem weiteren Schritt unsere HTML-Datei (cgisample.html) mit dem einfachen Formular.

```
<HTML>
  <HEAD>
    <TITLE>CGI-Sample</TITLE>
  </HEAD>

  <BODY>
    Click here to test CGI-Program.
    <FORM METHOD="POST" ACTION="/cgi-bin/helloworld.cgi">
    <INPUT TYPE="submit" VALUE="Start"/>
    </FORM>
  </BODY>
</HTML>
```

Abbildung 7: HTML-Datei cgisample.html

Nach dem Abspeichern sollte die Seite jetzt von einem Webbrowser durch die Eingabe ‚http://*serveradresse:portnummer*/firstSample/cgisample.html' abrufbar sein.

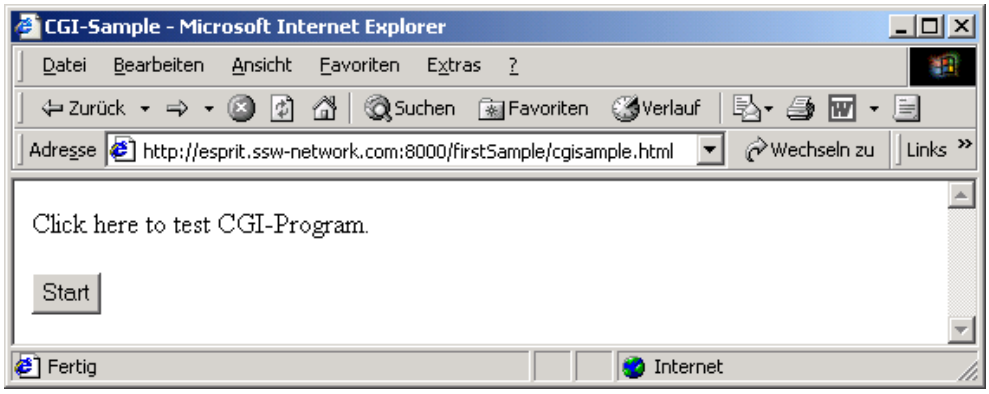

Abbildung 8: Einfaches HTML-Formular

! *ACHTUNG: Läuft Apache unter UNIX, sind alle Eingaben (auch die der URL) Case-Sensitiv! Bitte beachten Sie das auch beim Anlegen der Verzeichnisse!*

Natürlich passiert beim Klicken auf die Schaltfläche ‚Start' derzeit noch gar nichts (eigentlich wird eine Fehlermeldung ausgegeben). Deswegen wird im nächsten Schritt das CGI-Programm implementiert, das gestartet werden soll.

CGI-Programm

Apache erwartet CGI-Programme in einem bestimmten Verzeichnis. Das Verzeichnis wird bereits bei der Konfiguration (siehe oben) festgelegt, heißt aber standardmäßig ‚cgi-bin'. In diesem Verzeichnis muss sich dann das ausführbare CGI-Programm befinden. Unter der Verwendung des CGI Dev-Environments werden die CGI-Programme allerdings im entsprechenden ‚source'-Verzeichnis angelegt und später in das entsprechende Apache-Verzeichnis kopiert.

Im Unterverzeichnis ‚source' des Dev-Environment-Verzeichnisses erstellen wir eine neue C-Datei mit dem Namen ‚helloworld.c'.

```
#include <stdio.h>
#include "cgi-lib.h"
#include "html-lib.h"

int main() {
  /* write the header */
  html_header();
  html_begin("Hello World"); /* Set Title of HTML-Document */

  html_end();
  return 0;
}
```

Abbildung 9: helloworld.c

Dieses Programm verwendet bereits die Grundfunktionen von ‚cgihtml':

- html_header(): Diese Funktion erzeugt einen gültigen HTML-Header. Tatsächlich besteht der HTML-Header nur aus der Zeile ‚Content-type: text/html'.

- html_begin(): Diese Funktion erzeugt den ersten Teil des eigentlichen HTML-Dokuments. Der übergebene Parameter wird dabei automatisch zum Titel der HTML-Seite. Tatsächlich erzeugt die Funktion folgende Zeilen:
```
<html><head>
<title>Parameter_der_Funktion</title>
</head>

<body>
```

- html_end(): Diese Funktion erzeugt den Abschlussteil des eigentlichen HTML-Dokuments. Tatsächlich werden die beiden offenen Bezeichner ‚<body>' und ‚<html>' geschossen. Die Funktion erzeugt also folgende Zeile:
```
</body></html>
```

Das Programm selbst gibt also eine leere HTML-Seite zurück, die allerdings bereits über einen Titel verfügt. Dieser wird im Browser auch entsprechend in der Titelleiste angezeigt.

Wie aus diesem Beispiel ersichtlich, erzeugt html_begin() den Anfangsteil eines HTML-Dokuments währen html_end() den Schlussteil generiert. Zwischen diesen beiden Funktionen kann der Entwickler durch die Ausgabe beliebigen HTML-Codes eine gültige HTML-Seite erzeugen. Der HTML-Code wird dabei einfach auf die Standardausgabe mittels der in C existierenden Stanardroutinen (printf) geschrieben. Der Webserver sorgt dann dafür, dass diese Ausgabe entsprechend umgeleitet wird.

Übersetzen und Installieren

Das Programm selbst wird mit dem CGI Dev-Enviornment Kommando „buildcgi helloworld.c –o helloworld.cgi" übersetzt. Erfolgt die Übersetzung ohne Fehler, so kann das CGI Programm mit dem Dev-Environment Kommando „installcgi helloworld.cgi" am Webserver installiert werden.

Danach sollte es durch Anklicken der Schaltfläche „Start" in unserem HTML-Formular möglich sein, das CGI Programm zu starten.

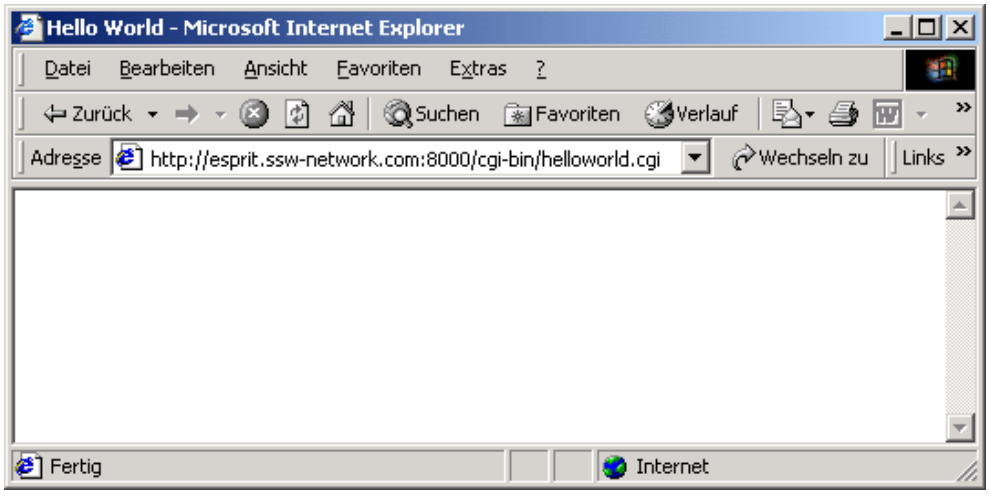

Abbildung 10: Ergebnis des CGI Programms „helloworld.cgi"

Dynamische HTML-Seiten Generierung

Dynamische HTML-Seiten Generierung

- Ausgaben auf Standardausgabe
  - CGI-Programm schreibt HTML-Code auf Standardausgabe
    - `printf("<H1>Ausgabe</H1>");`
  - Standardausgabe wird vom Webserver umgeleitet auf Client
- Testen: Aufruf von der Konsole
  - → HTML-Code wird am Bildschirm ausgegeben.

Bis jetzt haben wir gesehen, wie man mittels ‚cgihtml' eine leere HTML-Seite generieren kann. Es wurde bereits darauf hingewiesen, dass die Seite leicht mit Inhalten versehen werden kann, indem man einfach im CGI Programm HTML-Code auf die Standardausgabe schreibt. Das soll nun in einem weiteren Programm veranschaulicht werden.

Generieren von HTML-Code

Unser erstes CGI Programm soll nun so erweitert werden, dass das generierte Dokument die Nachricht ‚Hallo Welt, das ist ein erster CGI-Test' in großen Buchstaben enthält. Dazu modifizieren wir die C-Datei ‚helloworld.c' im Unterverzeichnis ‚source' des Dev-Environment-Verzeichnisses und erweitern diese um folgendes, fett dargestelltes Kommando:

```
#include <stdio.h>
#include "cgi-lib.h"
#include "html-lib.h"

int main() {
  /* write the header */
  html_header();
  html_begin("Hello World"); /* Set Title of HTML-Document */

  printf("<H1>Hallo Welt, das ist ein erster CGI-Test</H1>");

  html_end();
  return 0;
}
```

Abbildung 11: helloworld.c

Das Übersetzen und Installieren erfolgt genau gleich wie vorher, mittels Aufruf der Kommandos ‚buildcgi' und ‚installcgi'.

Klicken wir nun erneut auf die Schaltfläche „Start" in unserem Formular, erscheint nun ein dynamisch generiertes HTML-Dokument, das in großen Buchstaben unsere Begrüßungsnachricht enthält.

Abbildung 12: Ergebnis des CGI Programmaufrufs

Wie aus dem Programmbeispiel leicht ersichtlich, erfolgt die Generierung eines solchen Dokuments also durch einfaches Schreiben von HTML-Code an die Standardausgabe mittels des printf-Kommandos. Für den Programmierer wird so die Komplexität des Webservers abstrahiert. Zudem kann das Programm leicht getestet werden, da bei einem direkten Aufruf von der Kommandozeile die Ausgabe auf den Bildschirm erfolgt.

ⓘ Die Gestaltung der Web-Seite bleibt nun einzig und allein dem Programmierer überlassen und hängt von dessen HTML-Kenntnissen ab. Innerhalb des CGI-Programms kann auf die vollständige HTML-Syntax zugegriffen werden, genauso als ob man eine HTML-Seite mit einem beliebigen Texteditor erstellen würde.

Auswertung von HTML-Formularen

Auswertung von HTML-Formularen

- Daten werden vom Webserver zugestellt
- cgihtml parst Eingabe und überführt diese in eine verkettete Liste (Typ: `llist`)
  - `read_cgi_input(&liste);`
- Zugriff auf Werte der Liste
  - `cgi_val(liste, kontrollelement_name);→char*`
  - `cgi_val_multi(liste, kontrollelement_name);` `→ char**`
- Freigabe der Liste
  - `list_clear(&liste);`

Der Haupteinsatz von CGI Programmen ist die Auswertung von Daten, die in Form eines HTML-Formulars vom Benutzer eingegeben worden sind. Deswegen wird in weiterer Folge gezeigt, wie solche Formulardaten mit Hilfe von CGI Programmen ausgewertet werden können.

Zugriff auf Formulardaten

Um HTML-Formulardaten auswerten zu können, müssen diese zuerst entsprechend geladen und geparst werden. Dazu bietet ‚cgihtml' eine Reihe von Funktionen, die einerseits das Einlesen der CGI Daten erlauben, und andererseits den Zugriff auf bestimmte Daten der Formularkontrollelemente ermöglichen. Dazu bedient sich ‚cgihtml' einer Liste, in welche die einzelnen Daten abgelegt werden. Das entsprechende C-Konstrukt heißt ‚*llist*'. Die Funktion ‚`read_cgi_input(&liste)`' liest die CGI Daten vom Webserver und füllt eine *llist*-Liste mit den Formularwerten an. Der Zugriff auf die einzelnen Daten erfolgt mittels der Funktion ‚`cgi_val(liste, name_des_kontrollelements)`'. Diese Funktion liefert immer eine Zeichenkette, die den jeweiligen Formularwert des entsprechenden Kontrollelements enthält. Die Liste muss natürlich zum Schluss wieder freigegeben werden. Dies erfolgt durch den Aufruf der Funktion ‚`list_clear(&liste)`'. Die Funktionalität wird nun anhand eines weiteren Beispiels veranschaulicht.

Kontrollelemente Textbox und Optionsschalter

Um Daten eingeben zu können, muss das HTML-Formular einige Kontrollelemente bereitstellen. Dazu legen wir im htdocs-Verzeichnis des Webservers ein neues Unterverzeichnis mit dem Namen ‚formTest' an. In diesem Verzeichnis erzeugen wir mit Hilfe eines Texteditors eine neue HTML-Seite mit

dem Namen ‚formtest.html', die ein Formular mit einigen Kontrollelementen enthält.

```
<HTML>
  <HEAD>
    <TITLE>CGI Form Sample</TITLE>
  </HEAD>

  <BODY>
    Please fill out the form and click "Start" afterwards.
    <FORM METHOD="POST" ACTION="/cgi-bin/checkform.cgi">
      <INPUT TYPE="text" NAME="Textfield" SIZE="30"/><BR/>
      <INPUT TYPE="radio" NAME="RadioButtons"
        VALUE="Value1"/>Option 1
      <INPUT TYPE="radio" NAME="RadioButtons"
        VALUE="Value2"/>Option 2
      <INPUT TYPE="radio" NAME="RadioButtons"
        VALUE="Value3"/>Option 3<BR/>
      <INPUT TYPE="submit" VALUE="Start"/>
    </FORM>
  </BODY>
</HTML>
```

Abbildung 13: HTML-Datei formtest.html

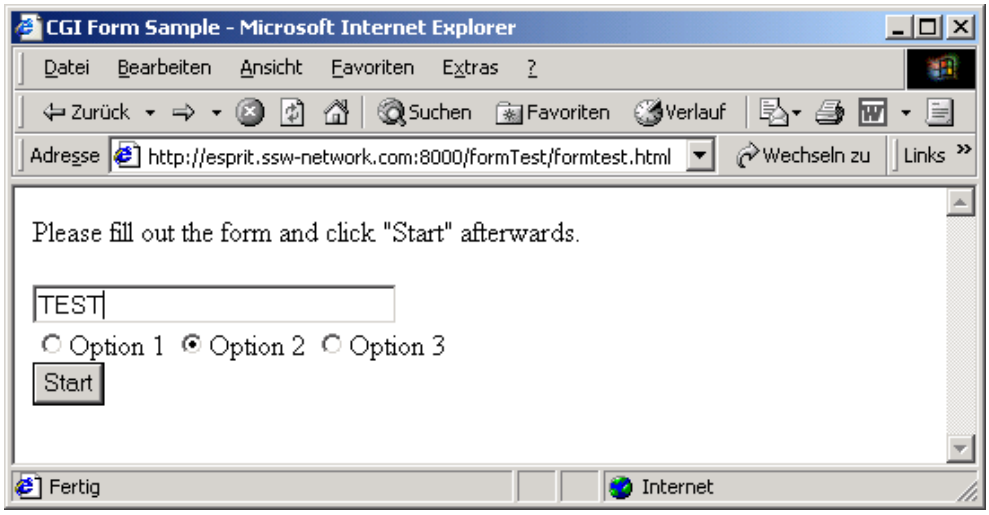

Abbildung 14: formtest.html im Webbrowser

Auswertung mittels eines CGI Programm

Die Daten können dann mittels eines CGI Programms ausgelesen werden. Dazu erstellen wir eine neue C-Datei im ‚source'-Verzeichnis des CGI Dev-Environment Verzeichnisses mit dem Namen ‚checkform.c'.

Die Funktionen zur Auswertung der CGI Daten sind im Programm fett dargestellt. Wie bereits erwähnt, werden die Daten in eine Liste des Typs ‚llist' abgelegt. Diese muss zuerst deklariert werden. Das eigentliche Einlesen der Daten erfolgt mittels der Funktion ‚read_cgi_input()', die eine solche Liste mit den Daten anfüllt. Auf die einzelnen Werte der Kontrollelemente kann dann mit der Funktion ‚cgi_val()' zugegriffen werden. Die Funktion erwartet als ersten Parameter die Datenliste und als zweiten Parameter den Namen des Kontrollelements, so wie er im HTML-Formular spezifiziert wurde. Als Ergebnis liefert die Funktion den Wert des Kontrollelements als Zeichenkette zurück. Zum Schluss muss die Liste mit der Funktion ‚list_clear()' freigegeben werden.

```
#include <stdio.h>
#include "cgi-lib.h"
#include "html-lib.h"

int main() {
  llist entries;   /* linked list containing all entries */

  /* now parse the form data and add it to the linked list */
  read_cgi_input(&entries);

  /* write the header */
  html_header();
  html_begin("Hello World"); /* Set Title of HTML-Document */

  printf("<H1>CGI Data</H1><BR/>");

  printf("The value entered in the textbox was: %s",
         cgi_val(entries, "Textfield"));
  printf("<BR/>");
  printf("The selected option was: %s",
         cgi_val(entries, "RadioButtons"));
  printf("<BR/>");

  html_end();
  list_clear(&entries);
  return 0;
}
```

Abbildung 15: checkform.c

Die Übersetzung und Installation erfolgt wieder mittels der Kommandos „buildcgi checkform.c –o checkform.cgi“ und „installcgi checkform.cgi“.

Wird das Programm nun durch das Klicken auf die Schaltfläche ‚Start’ im HTML-Formular gestartet, erscheint eine HTML-Seite mit der Auswertung der Formulardaten.

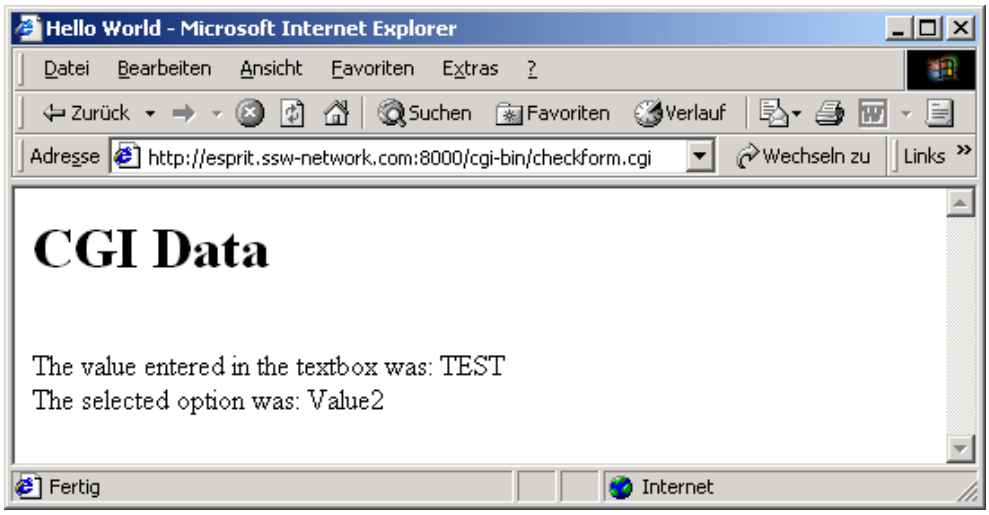

Abbildung 16: Ergebnis von checkform.cgi

Wie aus der Ergebnisseite deutlich hervorgeht, liefert die Funktion cgi_val() für Textboxen den eingegebenen Text, bei Optionsschalter den im HTML-Formular mittels des Attributs ‚VALUE’ angegebenen Wert der ausgewählten Option.

Erweiterung: Kontrollelement Auswahlliste

In Auswahllisten kann nur immer ein Element gewählt werden. Deswegen werden Auswahllisten genauso ausgewertet wie Textboxen oder Optionsschalter. Die Auswahlliste mit dem Namen ‚PopUp' würde wie folgt ausgewertet werden:

```
printf("The selected value in the Popup Box was: %s",
         cgi_val(entries, "PopUp"));
```

Abbildung 17: Auswertung einer Auswahlliste

Erweiterung: Kontrollelement Textbereichsbox

Auch Textbereichsboxen liefern den Inhalt als eine einzige Zeichenkette zurück. Zeilenumbrüche in Textboxen werden als Newline-Zeichen (‚\n') in der Ergebniszeichenkette angezeigt.

Gerade die Zeilenumbrüche sind problematisch, wenn eine Ausgabe der Textbereichsbox in einer HTML-Seite erfolgen soll, da das Newline-Zeichen vom Browser ignoriert wird. Um eine detailgetreue Ausgabe der Textbox in HTML zu ermöglichen, müssen Newline-Zeichen durch den Bezeichner ‚
' ersetzt werden. textArea = cgi_val(entries, "TextArea");

```
if (textArea != 0) {
  for (i = 0; textArea[i] != '\0'; i++) {
    if (textArea[i] == '\n')
      printf("<BR/>");
    else
      printf("%c", textArea[i]);
  }
}
```

Abbildung 18: Auswertung einer Textbox mit Newline-Konverzierung

Erweiterung: Kontrollelement Auswahlschalter

Im Gegensatz zu allen anderen, hier vorgestellten Kontrollelementen, können Auswahlschalter mehr als nur einen Wert zurückliefern. Wird deshalb der Wert der Auswahlschalter mit der Funktion ‚cgi_val()' ermittelt, hat man nur Zugriff auf die erste gewählte Option, allerdings werden *nicht alle* gewählten Optionen geliefert!

```
printf("The selected value in the Popup Box was: %s",
         cgi_val(entries, "PopUp"));
```

Abbildung 19: Auswertung von CheckBoxen mit cgi_val()

Abbildung 20 zeigt das Ergebnis einer solchen Auswertung. Wie in der Abbildung ersichtlich, wird für die Auswahlschalter nur der Wert der Option 2 geliefert, nicht aber der Wert der Option 3, welche im Formular aber ausgewählt wurde.

Für Kontrollelemente, die mehr als einen Wert zurückliefern können (z.B. Auswahlschalter) bietet ‚cgihtml' eine weitere Funktion namens ‚cgi_val_multi()' an. Diese Funktion erwartet genauso wie ‚cgi_val()' zwei Parameter, nämlich die CGI-Datenliste und den Namen des Kontrollelements. Allerdings liefert die Funktion ‚cgi_val_multi()' nicht einen Zeichenkettenwert (‚char*') zurück, sondern eine Liste von Zeichenketten (‚char**').

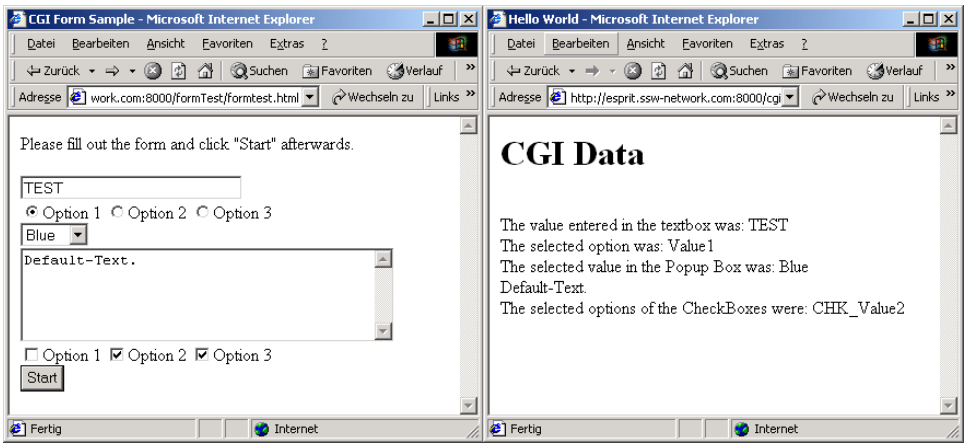

Abbildung 20: Auswertungsergebnis von CheckBoxen mit cgi_val()

Auswerten von Daten mittels ‚cgi_val_multi()'

Um Kontrollelemente mit mehreren möglichen Ergebnissen auswerten zu können, wird die Funktion ‚cgi_val_multi()' verwendet. Diese liefert als Rückgabewert eine Liste von Zeichenketten und benutzt dafür das Konstrukt ‚char**'. Mit Hilfe einer Schleife kann dann über diese Liste iteriert werden und auf die einzelnen Teilergebnisse zugegriffen werden.

```
char** result;
int    i;

...

result = cgi_val_multi(entries, "Name_des_Kontrollelements");

for (i=0; result[i] != NULL; i++) {
  printf("Option %s", result[i]);
  printf("<BR/>");
}
```

Abbildung 21: Auswerten von Kontrollelementen mit mehreren Ergebnissen

Ein zusammenhängendes Beispiel, welches alle Kontrollelemente und deren Auswertung enthält, findet sich im Anhang.

Diskussion

Diskussion

- Layout
  - cgihtml generiert selbst die Sektion <BODY> → Keine Einflussnahme auf Farbwahl
  - Angabe von Stylesheets wird nicht unterstützt
- Probleme von ‚cgihtml'
  - Bekannte Sicherheitsprobleme (Speicherverwaltung, temporäre Dateien)
- Mitführen von Zustandsinformationen
  - http ist Zustandslos (Eingaben werden vergessen)
  - Mitführen von Zuständen über
    `<INPUT TYPE="hidden"/>`

Layout von generierten Webseiten

Wie in diesem Tutorium gezeigt, erlaubt ‚cgihtml' das einfache generieren von HTML-Seiten. Allerdings wird kaum Augenmerk auf die graphische Gestaltung der Seite gelegt. So wird z.B. die Sektion ‚<BODY>' generiert, ohne Einfluss auf Farbwahl oder Hintergrundbilder zu nehmen. Auch das Einbeziehen von Style-Sheets wird von ‚cgihtml' nicht unterstützt. Sollen solche Webseiten realisiert werden, so muss die gesamte Webseite aus dem Programm mit Hilfe von Ausgaben generiert werden, ohne der Verwendung der Funktionen html_header(), html_begin(), und html_end().

Probleme von ‚cgihtml'

Bei der Verwendung von ‚cgihtml' sei darauf hingewiesen, dass eine Reihe von Problemen in Verbindung mit dieser Programmbibliothek bekannt sind. So sind eine Reihe von Sicherheitsproblemen bekannt, die schnell zu einem großen Problem werden können, wenn man bedenkt, dass CGI Programme ausschließlich am Server ausgeführt werden und somit den Server zum Absturz bringen können oder aber Sicherheitslücken ins Netzwerk eröffnen können.

Bekannte Probleme sind unsichere Verwendung von temporären Dateien, unsichere Speicherverwaltung und Anfälligkeit gegenüber absichtlich gefälschte Post-Daten die an das CGI Programm übermittelt werden.

Mitführen von Zustandsinformationen

Das im Web genutzte http-Protokoll ist ein zustandsloses Protokoll. Das bedeutet, dass Clientanfragen an Webseiten vom Server beantwortet werden, der Server sich aber nicht merkt, dass er mit einem bestimmten Client kommuniziert hat. Gerade bei der Implementierung von Online-Shops oder anderen Client-gesteuerten Webseiten ist das Mitführen von solchen Zustandsinformationen von Nöten. CGI selbst bietet keinerlei Mechanismen für das Mitführen von Zuständen, allerdings kann das Mitführen von Zuständen mittels spezieller Techniken in CGI realisiert werden.

Dazu benutzt man versteckte Kontrollelemente. Diese werden durch den Bezeichner ‚<INPUT TYPE="hidden">' realisiert. Mit Hilfe solcher Kontrollelemente können Informationen über mehrere, auch über CGI Programme generierte Seiten mitgeführt werden.

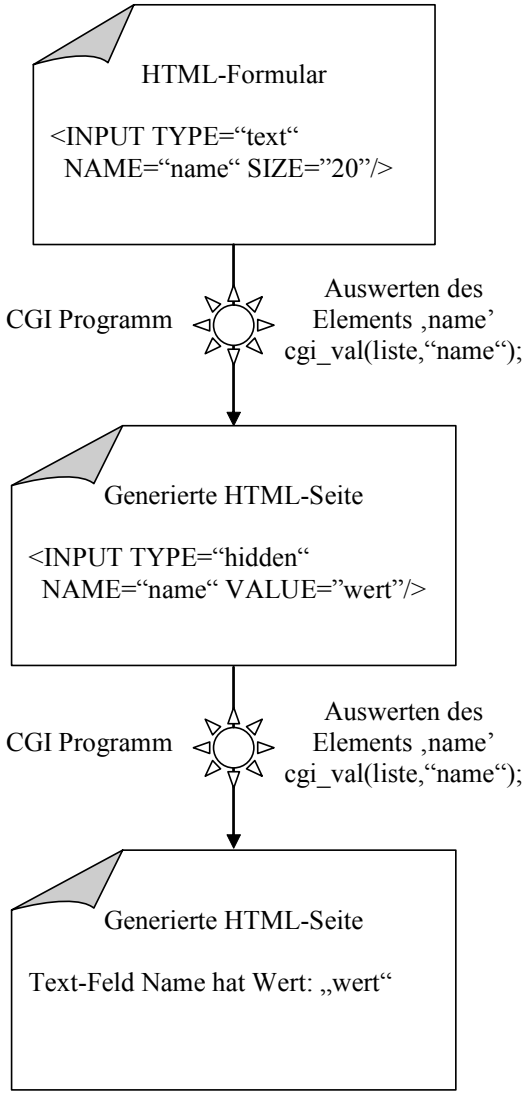

Abbildung 22: Konzept zum Mitführen von Zuständen in CGI Programmen

Anhang: Beispiel mit allen Kontrollelementen

Zwecks besseren Verständnisses wird nun ein durchgängiges Beispiel mit allen hier vorgestellten Kontrollelementen gezeigt.

HTML-Datei

```
<HTML>
  <HEAD>
    <TITLE>CGI Form Sample</TITLE>
  </HEAD>

  <BODY>
    Please fill out the form and click "Start" afterwards.
    <FORM METHOD="POST" ACTION="/cgi-bin/checkform.cgi">
      <INPUT TYPE="text" NAME="Textfield" SIZE="30"/><BR/>
      <INPUT TYPE="radio" NAME="RadioButtons"
        VALUE="Value1"/>Option 1
      <INPUT TYPE="radio" NAME="RadioButtons"
        VALUE="Value2"/>Option 2
      <INPUT TYPE="radio" NAME="RadioButtons"
        VALUE="Value3"/>Option 3<BR/>
      <SELECT NAME="PopUp" SIZE="1">
        <OPTION SELECTED>Blue</OPTION>
        <OPTION>Red</OPTION>
        <OPTION>Green</OPTION>
      </SELECT>
      <BR/>
      <TEXTAREA NAME="TextArea" ROWS=5 COLS=40>
        Default-Text.
      </TEXTAREA>
      <BR/>
      <INPUT TYPE="checkbox" NAME="CheckBoxes"
        VALUE="CHK_Value1"/>Option 1
      <INPUT TYPE="checkbox" NAME="CheckBoxes"
        VALUE="CHK_Value2"/>Option 2
      <INPUT TYPE="checkbox" NAME="CheckBoxes"
        VALUE="CHK_Value3"/>Option 3
      <BR/>
      <INPUT TYPE="submit" VALUE="Start"/>
    </FORM>
  </BODY>
</HTML>
```

CGI Programm

```
#include <stdio.h>
#include "cgi-lib.h"
#include "html-lib.h"

int main() {
  llist  entries;    /* linked list containing all entries */
  char*  textArea;
  char** result;
  int    i;

  /* now parse the form's data and add it to the list */
  read_cgi_input(&entries);

  html_header();
  html_begin("Hello World");   /* Set title of HTML doc */

  printf("<H1>CGI Data</H1><BR/>");

  printf("The value entered in the textbox was: %s",
         cgi_val(entries, "Textfield"));

  printf("<BR/>");
  printf("The selected option was: %s",
         cgi_val(entries, "RadioButtons"));

  printf("<BR/>");
  printf("The selected value in the Popup Box was: %s",
         cgi_val(entries, "PopUp"));

  printf("<BR/>");
  textArea = cgi_val(entries, "TextArea");

  /* Replace \n with <BR/> for HTML-Output */
  if (textArea != 0) {
    for (i = 0; textArea[i] != '\0'; i++) {
      if (textArea[i] == '\n')
        printf("<BR/>");
      else
        printf("%c", textArea[i]);
    }
  }

  printf("<BR/>");
  result = cgi_val_multi(entries, "CheckBoxes");

  /* Read multiple options of checkbox */
  for (i=0; result[i] != NULL; i++) {
    printf("Option %s", result[i]);
    printf("<BR/>");
  }

  html_end();
  list_clear(&entries);
  return 0;
}
```