CGI

Stand: 21.07.2022

Die CGI-Schnittstelle

Wenn ein serverseitiges Script angefordert wird, so muss der Webserver wissen, dass er es ausführen soll. Dazu bietet etwa der Apache Webserver zwei grundsätzliche Varianten an. Die eine besteht darin, den zugehörigen Script-Interpreter als Apache-Modul
einzubinden. Ein entsprechendes, die API-Schnittstelle des Apache bedienendes Modul muss natürlich existieren. So wird es in der heutigen Praxis mit PHP in der Regel gemacht.

Die zweite und ältere Variante ist die so genannte CGI-Schnittstelle. CGI steht für **Common Gateway Interface**. Die CGI-Schnittstelle besteht darin, dass bestimmte Verzeichnisse als CGI-Verzeichnisse definiert werden. In solchen Verzeichnissen
(und in Unterverzeichnisstrukturen davon) dürfen Scripts abgelegt werden, die der Webserver als für ihn ausführbar akzeptiert. In welcher Programmiersprache die Scripts geschrieben sind, ist dem Webserver egal. Wichtig ist nur, dass er den Script-Interpreter
ermitteln kann, z.B. über die Dateiendung der Scripts oder über deren Shebang-Zeile. Es kann sich natürlich auch um kompilierte Programme handeln, die ohne Interpreter direkt ausführbar sind.

Ein CGI-Script kann serverseitig Daten verarbeiten, mit Datenbanken kommunizieren usw.

Am Ende sollte es jedoch stets Daten auf die Standardausgabe schreiben. Da es vom Webserver ausgeführt wird, kann dieser die Ausgaben des Scripts auffangen und zum aufrufenden Browser senden. Bei seiner Antwort an den Browser beschränkt sich der Webserver
im HTTP-Protokoll auf das reine HTTP-Kommando ohne zusätzliche HTTP-Header. Diese muss das **CGI-Script** selber erzeugen. Der Vorteil davon ist eine große Flexibilität. CGI-Scripts erzeugen in der Regel HTML-Code, aber sie können ebenso
gut Binärdaten einer dynamisch erzeugten Grafik oder andere Daten ausgeben. Wichtig ist, dass die ausgegebenen HTTP-Header und die ausgegebenen Daten vom aufrufenden Browser, der die Daten vom Webserver gesendet bekommt, verarbeitbar sind. CGI-Scripts
erzeugen jedoch nicht nur Ausgaben, sondern verarbeiten meistens auch Input. Den Input erhalten sie beim Aufruf durch den Browser über die beiden möglichen Eingabekanäle GET oder POST. Bei GET werden die `Input`-Daten
bekanntlich als Query-String im URI mit übertragen. Bei POST werden sie über den Standardeingabekanal des Server-Rechners zur Verfügung gestellt.

CGI-Aufrufmöglichkeiten in HTML

Sobald Sie dieses Programm mittels der verschiedenen Aufrufmöglichkeiten ausführen, sorgen die Anweisungen dafür, dass der HTML Quelltext zur Interpretation und Ausgabe an den Browser zurückgegeben wird. In HTML gibt es verschiedene Orte, an denen ein

CGI-Script aufgerufen werden kann:

- **Über ein Formular**: Beispiel: `<form action="/cgi-bin/guestbook.pl" method="post">`. Durch Absenden des Formulars sendet der Browser an den Webserver über HTTP im Beispiel eine POST-Anforderung für die
  Datenressource /cgi-bin/guestbook.pl. Der Webserver erkennt, dass es sich um ein CGI-Script handelt, und führt es aus. Die vom Anwender eingegebenen oder ausgewählten Daten stehen dem CGI-Script zur Verarbeitung zur Verfügung – im obigen
  Beispiel über den POST-Eingabekanal. Formulare sind auch die einzige Möglichkeit in HTML, CGI-Scripts über den POST-Eingabekanal mit Daten zu versorgen.
- **Über Verweise**: Es genügt, als URI des Verweisziels das ausführbare CGI-Script anzugeben, z.B. `<a href="/cgi-bin/statistik.pl">Statistik</a>`. Dies ist vor allem sinnvoll für CGI-Scripts, die keinen Input vom
  Anwender benötigen, sondern lediglich feste Datenausgaben erzeugen, zum Beispiel für ein CGI-Script, das aktuelle Zugriffsstatistiken für Webseiten ausgibt.
- **Über eine Grafikreferenz**: Auch dabei genügt es, als URI in der `src`-Angabe des `<img>`-Tags das ausführbare CGI-Script anzugeben, z.B. `<img src="/cgi-bin/counter.pl">`. Dabei muss das CGI-Script allerdings eine Grafikdatei im GIF- oder JPEG-Format an den Webbrowser zurücksenden. Die meisten grafischen Zugriffszähler basieren auf diesem Prinzip.
- **Über Server Side Includes**: Über eine SSI-Anweisung in einer SHTML-Datei wie z.B. `<!-- #exec cgi="/cgi-bin/counter.pl" -->` kann ein CGI-Script automatisch beim Aufruf der SHTML-Datei ausgeführt werden. Das ist sehr
  praktisch, um mithilfe eines CGI-Scripts dynamische Information in Textform in eine HTML-Datei einzubinden.

CGI-Konfiguration

Welche Verzeichnisse als **CGI-Verzeichnisse** gelten, wird in der Konfiguration des Webservers festgelegt. Beim Apache Webserver kann dies wahlweise in der zentralen Konfiguration der Datei httpd.conf geschehen oder in verzeichnisspezifischen
.htaccess-Dateien.

Zentrales CGI-Verzeichnis einrichten

Um in der httpd.conf ein zentrales CGI-Verzeichnis zu bestimmen, müssen Sie die Direktive `ScriptAlias` verwenden. Suchen Sie in der httpd.conf nach einer entsprechenden Direktive. Per Default ist bereits eine Direktive
für CGI-Scripts eingetragen, jedoch auskommentiert. Entfernen Sie die Auskommentierung, also das führende Gatterzeichen (#). Ein typischer Eintrag lautet:

```
ScriptAlias /cgi-bin/ "/var/www-scripts/cgi-bin/"
```

Info Die Direktive erwartet zwei Parameter. Im ersten Parameter wird der absolute Pfadname aus URI-Sicht

angegeben und im zweiten Parameter der absolute Pfadname aus Sicht des Dateisystems am Server-Rechner. Der zweite Parameter steht in hohen Anführungszeichen.
Beachten Sie, dass beide Pfadnamen am Ende noch mal einen Schrägstrich enthalten müssen!

> Das CGI-Verzeichnis muss übrigens nicht innerhalb der Document Root liegen.

Angenommen, im obigen Beispiel wurde DocumentRoot /var/www festgelegt. Dann befindet sich das angegebene CGI-Verzeichnis parallel dazu auf gleicher Verzeichnisebene, aber nicht unterhalb davon. Der Webserver löst Client-Anfragen wie http://server/cgi-bin/test.cgijedoch auf Grund der Zuordnung bei der `ScriptAlias` korrekt auf. Damit **CGI-Scripts** funktionieren, muss beim Apache ferner das CGI-Modul geladen werden. Dazu darf in der httpd.conf der Eintrag, der mit `LoadModule cgi_module`... beginnt, nicht auskommentiert sein. Sollte dies der Fall sein, müssen Sie das Kommentargatterzeichen entfernen. In jedem Fall ist es ferner sinnvoll, in folgender Zeile einer httpd.conf das Kommentarzeichen zu entfernen:

```
AddHandler cgi-script .cgi
```

Damit dürfen CGI-Scripts die Dateiendung *.cgi* haben.

Dezentral CGI-Verzeichnisse einrichten

Die zweite Möglichkeit besteht darin, mehrere CGI-Verzeichnisse einzurichten. Das kann entweder innerhalb einer Direktivengruppe zwischen <Directory> und </Directory> geschehen oder innerhalb des gewünschten Verzeichnisses
selbst in einer .htaccess-Datei. In letzterem Fall muss sich das Verzeichnis allerdings innerhalb der **Document Root** befinden. In beiden Fällen können Sie folgende Direktiven notieren:

```
Options +ExecCGI
AddHandler cgi-script .cgi
```

CGI-Scripts in allen Webverzeichnissen erlauben

Es ist auch möglich, CGI-Scripts in allen Verzeichnissen unterhalb der Document Root zu erlauben. Dazu müssen Sie in der httpd.conf die Direktivengruppe `<Directory>...</Directory>` für das Verzeichnis suchen,
das als Document Root definiert wurde. Wenn Sie dort bei der Direktive `Options` den Wert `ExecCGI` mit angeben, können CGI-Scripts in allen Webverzeichnissen abgelegt werden.

Voraussetzungen für CGI-Scripts

Die wesentliche Voraussetzung ist dabei die, dass der Server weiß, aus welchen Verzeichnissen er heraus CGI-Scripts ausführen soll und aus welchen nicht. Auch die CGI-Scripts selber müssen Voraussetzungen erfüllen, damit die CGI-Schnittstelle tatsächlich

bedient werden kann:

- Das Script muss in einem als CGI-Verzeichnis konfigurierten Verzeichnis abgelegt werden.
- Sowohl das CGI-Verzeichnis als auch das CGI-Script müssen auf einem Linux-/Unix- System für den Webserver ausführbar sein. Da der **Webserver** als ganz normaler Benutzer vom Typ „Rest der Welt“ auf dem System läuft, müssen die Ausführrechte
  entsprechend gesetzt werden. Am sinnvollsten ist es, wenn sowohl Dateieigentümer als auch seine Primärgruppe und der Rest der Welt Ausführrechte erhalten. Das Schreibrecht sollte jedoch dem Eigentümer oder maximal noch seiner Gruppe vorbehalten
  sein. In der Regel ist `chmod 755` für das CGI-Verzeichnis und die Scripts darin eine sinnvolle Wahl.
- Das Script muss zumindest einen HTTP-Header für das Feld `CONTENT-TYPE` ausgeben.

CGI-Umgebungsvariablen

Zur so genannten „CGI-Schnittstelle“ gehören auch die **CGI-Umgebungsvariablen**. Ein installierter Webserver stellt solche Variablen auf Betriebssystemebene zur Verfügung. Bei jedem Aufruf eines CGI-Scripts füllt er einige dieser Variablen
mit Informationen. Das CGI-Script kann den Inhalt dieser Variablen auslesen und für eigene Zwecke nutzen. Das CGI-Script kann einige dieser Variablen auch selbst mit Inhalt füllen oder einen dort gespeicherten Inhalt ändern. CGI-Umgebungsvariablen
existieren unabhängig von den Variablen, die Sie in einem CGI-Script selbst definieren können.

In CGI-Umgebungsvariablen ist während der Ausführung eines CGI-Scripts beispielsweise der Namen des Server-Rechners gespeichert oder Information zum verwendeten Browser des Anwenders, der den Aufruf des CGI-Programms verursacht hat. Wenn zum Aufruf des
CGI-Scripts aus einem Formular die GET-Methode verwendet wurde, stehen die Daten, die der Anwender in das Formular eingegeben hat, in einer CGI-Umgebungsvariable.

> Um CGI-Umgebungsvariablen aus einem CGI-Programm heraus zu nutzen,

müssen Sie in der verwendeten Programmiersprache die Techniken zum Auslesen von Umgebungsvariablen einsetzen. In Perl können Sie beispielsweise mit einer Anweisung wie `print $ENV{'SERVER_NAME'};` den Hostnamen des Servers ausgeben. Perl stellt
nämlich alle verfügbaren Umgebungsvariablen in einem vordefinierten Hash namens `%ENV;` zur Verfügung. Um herauszufinden, welche Umgebungsvariablen auf Ihrem Webserver verfügbar sind, können Sie das folgende kleine Perl-Script verwenden. Den
Quellcode können Sie mit einem Texteditor im eingestellten CGI-Verzeichnis beispielsweise unter dem Namen env.pl abspeichern. Angenommen, Vorausgesetzt, Ihr CGI-Verzeichnis heißt wie üblich cgi-bin und der Webserver ist gestartet,
dann können Sie das **Script** im Browser mit der Adresse http://127.0.0.1/cgi-bin/env.pl oder auch mit der Adresse http://localhost/cgi-bin/env.pl aufrufen. Datei **env.pl**

```
#!/usr/bin/perl -w
```

```
use strict;
use CGI::Carp qw(fatalsToBrowser);
print "Content-type: text/html\n\n";
print '<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">', "\n";
print "<html><head>
    <title>Umgebungsvariablen</title></head><body>\n";
print "<h1>Umgebungsvariablen:</h1>\n";
print "<table border=\"1\">\n";
print "<tr><th align=\"left\" bgcolor=\"#E0E0E0\">Variablenname</th>",
    "<th align=\"left\" bgcolor=\"#E0E0E0\">Wert</th></tr>\n";
foreach(keys( % ENV)) {
    print "<tr><td><b>$_</b></td>
        <td><tt>$ENV {
            $_
        } </tt></td>
       </tr>\n";
}
print "<tr><th align=\"left\" bgcolor=\"#E0E0E0\" colspan=\"2\">insgesamt: ",
    scalar keys(%ENV)," Umgebungsvariablen</th></tr>\n";
print "</table>\n";
print " </body></html>\n";
```

Info Das Script erzeugt eine HTML-Tabelle mit allen auf dem Server verfügbaren Umgebungsvariablen. Auf die Perl-Anweisungen wird an dieser Stelle noch nicht näher eingegangen. Nur die erste Zeile des Scripts (die so genannte shebang) müssen Sie möglicherweise
anpassen. Dort, wo im Beispiel `#!/usr/bin/perl` steht, müssen Sie, falls Sie beim Aufruf eine Fehlermeldung wie Internal Server Error erhalten, den kompletten Pfad zur ausführbaren Datei, also zum Perl-Interpreter, angeben. Wenn Sie den **Perl-Interpreter**unter MS Windows beispielsweise unter c:\programme\perl installiert haben, lautet Ihre erste Script-Zeile also:

```
#!c:/programme/perl/bin/perl
```

Die Dateinamen-Erweiterung *.exe* können Sie ohne weiteres weglassen.

Übersicht bekannter CGI-Umgebungsvariablen

Die folgende Tabelle ist nach Variablennamen alphabetisch sortiert.

Variablenname	Erläuterung
CONTENT_LENGTH	Enthält die Anzahl der Zeichen, die beim Aufruf des CGI-Scripts über die POST-Methode übergeben wurden. Wenn das CGI-Script beispielsweise beim Absenden eines HTML-Formulars aufgerufen wurde und dort als übertragungsmethode POST angegeben ist, steht in dieser Umgebungsvariablen, wie viele Zeichen das Script von der Standardeingabe lesen muss, um die übermittelten Formulardaten vollständig einzulesen.

Variablenname	Erläuterung
CONTENT_TYPE	Enthält beim Aufruf über die POST-Methode den MIME-Typ der übergebenen Daten. Wenn das CGI-Script beispielsweise beim Absenden eines HTML-Formulars aufgerufen wurde und dort als übertragungsmethode POST angegeben ist, steht in dieser Umgebungsvariablen der für HTML-Formulare typische MIME-Typ application/x-www-form-urlencoded (zu diesem MIME-Typ siehe auch Datenstrom bei übertragung von Formulardaten).
DOCUMENT_ROOT	Enthält den physischen Pfad des Wurzelverzeichnisses für die Ablage von Dateien, die im Webserver aufrufbar sind. Ein CGI-Script kann aus dieser Angabe beispielsweise absolute Pfadnamen zum öffnen von Dateien errechnen.
GATEWAY_INTERFACE	Enthält die Version der CGI-Schnittstelle, die von dem installierten Server unterstützt wird, z.B. CGI/1.1, wenn die gegenwärtig übliche Version 1.1 der Schnittstellendefinition unterstützt wird.
HTTP_ACCEPT	Enthält die Liste der MIME-Typen, die der aufrufende Web-Browser akzeptiert. Die Angabe */* bedeutet: der Web-Browser akzeptiert alles.
HTTP_ACCEPT_CHARSET	Enthält die Liste der Zeichenkodierungen, die der aufrufende Web-Browser akzeptiert, beispielsweise iso-8859-1, utf-8, utf-16, *;q=0.1.
HTTP_ACCEPT_ENCODING	Enthält eine Liste der Kodierungsmethoden, die der aufrufende Browser akzeptiert. Manche Browser akzeptieren beispielsweise auch den Kodierungstyp gzip, was bedeutet, dass der Browser auch Dateien empfangen kann, die nach dem GNU-Zip-Algorithmus komprimiert an ihn übertragen werden.
HTTP_ACCEPT_LANGUAGE	Enthält, welche Landessprache der aufrufende Browser bei seiner Benutzeroberfläche verwendet. Häufige Werte sind z.B. de (für deutschsprachige Browser) oder en (für englischsprachige Browser). Ein CGI-Script kann aufgrund dieser Angabe beispielsweise entscheiden, ob es eine deutschsprachige oder eine englischsprachige Antwort an den Browser sendet.
HTTP_CONNECTION	Enthält Informationen über den Status der HTTP-Verbindung zwischen Server und aufrufendem Browser. Der Wert Keep-Alive bedeutet, der Browser wartet auf Antwort.
HTTP_COOKIE	Enthält Namen und Wert von Cookies, sofern solche vom aufrufenden Browser gesendet werden. Mit der Perl-Anweisung: my @cookies = split(/[;,]\s*/,$ENV{‚HTTP_COOKIE‘}); können Sie alle gesetzten Cookies ermitteln. Jedes Element des Arrays namens @cookies enthält dann jeweils einen Cookie, bestehend aus einem Namen und einem Wert, die durch ein Gleichheitszeichen = getrennt sind. Der Wert eines Cookies ist im Format des MIME-Typs application/x-www-form-urlencoded gespeichert (zu diesem MIME-Typ siehe auch Datenstrom bei übertragung von Formulardaten).
HTTP_HOST	Enthält den Domain-Namen oder die IP-Adresse aus der Adresszeile des aufrufenden Browsers. Für ein CGI-Script kann diese Angabe wichtig sein, falls es mehrere Server bedienen muss.

Variablenname	Erläuterung
HTTP_REFERER	Enthält den URI der Web-Seite, von der aus das CGI-Script aufgerufen wurde. Der Wert wird jedoch nicht von allen Web-Browsern korrekt übermittelt, ist also nicht in jedem Fall verfügbar. Anmerkung: Wenn Ihnen die englische Sprache geläufig ist, werden Sie feststellen, dass HTTP_REFERER eigentlich HTTP_REFERRER heißen müsste. Den Autoren der ersten Spezifikation von HTTP ist dieser Fehler unterlaufen, der sich aus historischen Gründen und aufgrund der Abwärtskompatibilität von HTTP bis heute hält.
HTTP_USER_AGENT	Enthält Produkt- und Versionsinformationen zum aufrufenden Web-Browser. Ein CGI-Script kann auf diese Weise ermitteln, welchen Browser ein Anwender verwendet.
PATH_INFO	Wird einem CGI-Script eine Zeichenkette mit Daten übergeben, dann enthält PATH_INFO den Teil der Zeichenkette nach dem Namen des Scripts bis zum ersten ?. Wenn das Script beispielsweise die Adresse http://meine.seite.net/cgi-bin/test.pl hat, aber mit http://meine.seite.net/cgi-bin/test.pl/querys/musicbase.sql?cat=Mozart aufgerufen wird, dann enthält diese Umgebungsvariable den Anteil /querys/musicbase.sql. Sie ist dazu gedacht, Dateinamen mit Pfadangabe als übergabeparameter für Scripts zu ermöglichen.
PATH_TRANSLATED	Enthält wie PATH_INFO den Anteil des URI nach dem Scriptnamen bis zum ersten ?, jedoch mit dem Unterschied, dass nicht der Anteil selbst aus dem URI zurückgegeben wird, sondern der vom Webserver übersetzte Datenpfad dieses Anteils. Angenommen, das Script hat die Adresse http://meine.seite.net/cgi-bin/test.pl, wurde aber mit http://meine.seite.net/cgi-bin/test.pl/querys/musicbase.sql aufgerufen. Dann könnte der zusätzliche Adressanteil /querys/musicbase.sql aus Sicht des Webservers beispielsweise in einen physischen Pfadnamen wie /usr/web/seite/querys/musicbase.sql aufgelöst werden. Diesen Pfadnamen würde PATH_TRANSLATED zurückgeben.
QUERY_STRING	Enthält eine Zeichenkette mit Daten, die dem Script im URI nach dem ersten ? übergeben wurden. Angenommen, das Script hat die Adresse http://meine.seite.net/cgi-bin/test.pl, wurde aber mit http://meine.seite.net/cgi-bin/test.pl?User=Stefan aufgerufen. Dann würde QUERY_STRING den Wert User=Stefan enthalten. Wenn ein Anwender ein HTML-Formular ausgefüllt hat, bei dessen Absenden das CGI-Script mit der GET-Methode aufgerufen wurde, dann stehen in dieser Umgebungsvariablen die ausgefüllten Formulardaten. Die Daten sind nach den Regeln des MIME-Typs application/x-www-form-urlencoded kodiert.
REMOTE_ADDR	Enthält die IP-Adresse des Server-Rechners, über den das CGI-Script aufgerufen wurde. Es muss sich hierbei nicht unbedingt um die IP-Adresse des aufrufenden Client-Rechners handeln – der Wert kann beispielsweise auch von einem Proxy-Server stammen.
REMOTE_HOST	Enthält den Hostnamen des Rechners, über den das CGI-Script aufgerufen wurde. Dieser Wert wird jedoch nur gesetzt, wenn der Webserver entsprechend konfiguriert und dazu in der Lage ist, der IP-Adresse den entsprechenden Hostnamen zuzuordnen. Es muss sich hierbei nicht unbedingt um die IP-Adresse des aufrufenden Client-Rechners handeln – der Wert kann beispielsweise auch von einem Proxy-Server stammen.
REMOTE_IDENT	Enthält Protokollinformationen, wenn auf dem Server das Protokoll ident für geschützte Zugriffe läuft.

Variablenname	Erläuterung
REMOTE_PORT	Ermittelt, über welchen Port des Client-Rechners das CGI-Script aufgerufen wurde. Diese Zahl liegt gewöhnlich im Bereich ab 1024 aufwärts und wird vom aufrufenden Web-Browser zufällig ausgewählt.
REMOTE_USER	Enthält den Benutzernamen, mit dem sich der aufrufende Benutzer angemeldet hat, um das CGI-Script ausführen zu lassen. Wenn das Script beispielsweise htaccess-geschützt ist, muss sich der aufrufende Benutzer mit Benutzernamen und Passwort anmelden. Der dabei eingegebene Benutzername kann mit dieser Variable ermittelt werden.
REQUEST_METHOD	Enthält die HTTP-Anfragemethode, mit der das CGI-Programm aufgerufen wurde. Beispielsweise GET oder POST. Ein CGI-Script kann diese Variable auslesen und danach entscheiden, wie es Formulardaten einlesen kann: entweder von der Standardeingabe (bei Methode POST) oder aus der Umgebungsvariablen QUERY_STRING (bei Methode GET).
REQUEST_URI	Enthält den HTTP-Pfad des Scripts inklusive der im Aufruf übergebenen Daten. Angenommen, das Script hat die Adresse http://meine.seite.net/cgi-bin/test.pl und wurde mit http://meine.seite.net/cgi-bin/test.pl?User=Stefan aufgerufen. Dann liefert REQUEST_URI den Wert /cgi-bin/test.pl?User=Stefan.
SCRIPT_FILENAME	Enthält den physischen Pfad des Scripts auf dem Server-Rechner, also z.B. /usr/web/data/cgi-bin/test.pl.
SCRIPT_NAME	Enthält den HTTP-Pfad des Scripts. Angenommen, das Script hat die Adresse http://meine.seite.net/cgi-bin/test.pl. Dann liefert SCRIPT_NAME den Wert /cgi-bin/test.pl.
SERVER_ADDR	Enthält die IP-Adresse des Server-Rechners.
SERVER_ADMIN	Enthält Namen/E-Mail-Adresse des in der Webserver-Konfiguration eingetragenen Server-Administrators.
SERVER_NAME	Enthält den Namen des Server-Rechners, auf dem das CGI-Script läuft. Normalerweise ist dies der eingetragene Hostname des Rechners.
SERVER_PORT	Enthält die Portnummer, die für den Webserver eingerichtet wurde. Normalerweise ist dies für Webserver die Nummer 80.
SERVER_PROTOCOL	Enthält die Version des HTTP-Protokolls, das der installierte Webserver unterstützt, z.B. HTTP/1.1, wenn die gegenwärtig übliche Version 1.1 des HTTP-Protokolls unterstützt wird.
SERVER_SIGNATURE	Enthält eine erweiterte Selbstauskunft des Servers, z.B. Apache/1.3.31 Server at localhost Port 80.
SERVER_SOFTWARE	Enthält den Namen und die Versionsnummer der Webserver-Software auf dem Server-Rechner.

Beispiel eines Form-Mailers in Perl

In einem kleinen Praxisbeispiel möchten wir den typischen Zusammenhang zwischen HTML und CGI-Script demonstrieren. Das Beispiel realisiert einen so genannten **Form-Mailer**. Das ist die ideale Anwendung, wenn vom Anwender eingegebene Formulardaten
per Mail an den Site-Betreiber gesendet werden sollen. Der Anwender erhält nach Absenden des Formulars eine freundliche Danke-fürs-Ausfüllen-Seite am Bildschirm. Seine eingegebenen Daten werden zu einer ordentlich formatierten E-Mail aufbereitet und
an die gewünschte Mailadresse des Site-Betreibers gesendet.

> Das Beispiel ist in Perl geschrieben, da Perl immer noch die verbreitetste Sprache für klassische CGI-Scripts ist.

Auf den meisten bei Hosting-Providern zu mietenden Root-Servern ist Perl auch schon vorinstalliert, so dass Sie den Perl-Interpreter nicht noch installieren müssen. Sollte Perl nicht installiert sein, was Sie am Shell-Prompt sehr einfach durch das Kommando
`which perl` herausbekommen können, dann müssen Sie es downloaden und auf dem Server-Rechner installieren.

Perl kann von perl.com bezogen werden und ist ebenso wie Linux, Apache oder PHP OpenSource.

Für unser Form-Mail-Beispiel benötigen wir drei Dateien:

- Eine HTML-Datei mit dem Formular
- Eine HTML-Datei, die als Danke-Seite fungiert
- Das Perl-Script, das die Daten als CGI-Script verarbeitet und die Danke-Seite sendet

Teil 1: HTML-Datei mit Formular

Im Beispiel soll ein typisches **Feedback-Formular** realisiert werden. Die Datei mit dem Feedback-Formular ist eine gewöhnliche statische HTML-Datei. Der Quelltext lautet:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Feedback</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<h1>Feedback</h1>
<form action="/cgi-bin/form-mail.cgi" method="post">
<input type="hidden" name="return" value="/danke.htm">
```

```
<input type="hidden" name="delimiter" value=": ">
<input type="hidden" name="subject" value="Feedback-Formular">
<fieldset>
<legend>Hinweis:</legend>
<div class="explain">Felder mit Stern * müssen ausgefüllt werden.</div>
</fieldset>
</td>
</tr>
<tr>
<td>
<fieldset>
<legend>Daten:</legend>
<div>Vorname:
<br>
<input type="text" name="Vorname" class="text" style="width:400px">
</div>
<div>Zuname:
<br>
<input type="text" name="Zuname" class="text" style="width:400px">
</div>
<div>E-Mail: <b>*</b>
<br>
<input type="text" name="Mail" class="text" style="width:400px">
</div>
<div>Feedback: <b>*</b>
<br>
<textarea name="Text" class="text" style="width:400px; height:300px">
</textarea>
</div>
</fieldset>
<fieldset>
<legend>Daten:</legend>
<p>
<input type="submit" class="button" value="Absenden">
<input type="reset" class="button" value="Löschen">
</p>
</fieldset>
</form>
</body>
</html>
```

Info Layout und Optik der Seite und des Formulars werden in der eingebundenen Datei styles.css definiert. Darauf gehen wir an dieser Stelle nicht weiter ein. Wichtig ist im aktuellen Zusammenhang nur das Formular selbst. Im einleitenden `<form>`-Tag
wird im `action`-Attribut das **CGI-Script** aufgerufen, das die Daten nach Absenden des Formulars verarbeitet. Die sichtbaren Formularfelder sind frei benennbar, Art und Anzahl der Felder ebenfalls. Denn unser Form-Mail-Script

ist so geschrieben, dass es alle Arten von Formularen verarbeitet. Der Form-Mailer erwartet jedoch bestimmte `hidden`-Felder. Die Werte der `name`-Attribute dieser Felder sind durch das Form-Mail-Script vorgegeben. Die Werte (`value`)
müssen Sie dagegen im HTML-Code anpassen.

```
<input type="hidden" name="return" value="/danke.htm">
```

Damit geben Sie den URI der Seite an, die nach erfolgreichem Versenden des Formulars und Verarbeitung an den Seitenbesucher gesendet werden soll.

```
<input type="hidden" name="delimiter" value=":">
```

Damit geben Sie an, wie Feldnamen und Feldinhalte in der Mail, die versendet wird, dargestellt werden sollen. Im Beispiel stehen in den Mails, die das CGI-Script erzeugt, Zeilen wie: Feldname: vom Benutzer eingegebener Wert

```
<input type="hidden" name="subject" value="Feedback-Formular">
```

Damit bestimmen Sie das Subject der Mails, die vom CGI-Script versendet werden.

Teil 2: Danke-Seite

Bevor wir auf das Form-Mail-Script kommen, noch kurz der Beispielcode für die Danke-Seite. Hierbei ist aus technischer Sicht nichts Besonderes zu beachten. Es kann sich um eine normale statische HTML-Datei handeln. Die Seite sollte den inhaltlichen Bezug
zum gerade abgesendeten Formular herstellen. In unserem Beispiel lautet der Quelltext:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="de">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
<title>Danke für Ihr Feedback</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<h1>Danke für Ihr Feedback</h1>
<p>Ihre eingegebenen Daten wurden uns zugesendet. Wir werden uns mit Ihnen in
Verbindung setzen.</p>
</body>
</html>
```

Teil 3: Perl-Script für Form-Mailer

Das CGI-Script ist in unserem Beispiel der serverseitige Vermittler zwischen der Seite mit dem Feedback-Formular und der Danke-Seite. Nach dem Absenden des Formulars sendet der Browser des Anwenders einen POST-Request an den Webserver. Einen
POST-Request deshalb, weil im **HTML-Formular** als Übertragungsmethode `type="hidden"` angegeben wurde. Daraus ergeben sich folgende Aufgaben für das CGI-Script:

- Es muss die Formulardaten über den POST-Eingabekanal einlesen.
- Es muss aus den Formulardaten eine E-Mail zusammenstellen.
- Es muss die E-Mail an die gewünschte Mailadresse versenden.
- Es muss die Danke-Seite ausgeben. Der Webserver sendet diese dann an den aufrufenden Browser.

Der Quelltext unseres Perl-Scripts lautet:

```
#!/usr/bin/perl

$mailto = "IhrName@irgendwo";
$Sendmail_Prog = "/usr/lib/sendmail";
use CGI;
$query = new CGI;@
names = $query - > param;#
interne Daten aus den erwarteten hidden - Feldern auslesen:
    $delimiter = $query - > param('delimiter');#
-- - > Begrenzerzeichen
$returnhtml = $query - > param('return');#
-- - > Dankeseite
$subject = $query - > param('subject');#
-- - > E - Mail - Subject# Text der E - Mail aus den Formulardaten ermitteln:
    $mailtext = "";
foreach(@names) {
    $name = $_;@
    values = "";@
    values = $query - > param($name);
    if ($name ne "mailto" && $name ne "return" && $name ne "subject" && $name ne
"delimiter") {
        foreach $value(@values) {
            $mailtext = $mailtext.$name;
            $mailtext = $mailtext.$delimiter;
            $mailtext = $mailtext.$value.
            "\n";
        }
    }
}#
E - Mail versenden:
    open(MAIL, "|$Sendmail_Prog -t") ||
    print STDERR "Mailprogramm nicht gestartet werden\n";
```

```
print MAIL "To: $mailto\n";
print MAIL "Subject: $subject\n\n";
print MAIL "$mailtext\n";
close(MAIL);#
Dankeseite an Browser senden:
    print "Location: $returnhtml\n\n";
```

Info In der Shebang-Zeile wird der Pfad zum **Perl-Interpreter** angegeben. Üblicherweise ist die ausführbare Datei des Perl-Interpreters auf Linux-Systemen unter /usr/bin/perl erreichbar. Andernfalls müssen Sie hier den Pfadnamen
anpassen. Aus Sicht von Perl ist das eine Kommentarzeile, da sie mit einem Gatterzeichen (#) beginnt. Im Script-Code selber müssen Sie dann noch die erste Zeile anpassen:

```
$mailto = "IhrName@irgendwo";
```

Info Als Mailadresse ist hier diejenige Adresse einzusetzen, an die Daten abgesendeter Formulare gesendet werden sollen. Es hat übrigens einen sehr wichtigen Grund, dass wir diese Adresse erst im Form-Mail-Script konfigurieren und nicht schon in einem
`hidden`-Feld im HTML-Formular. Der Grund ist, dass wir durch die Definition in HTML ein öffentlich aufrufbares CGI-Script hätten, dass beliebige Inhalte an beliebige E-Mail-Adressen senden kann, wenn es nur per POST-Methode entsprechenden
Input erhält. Solch ein Script würde einen erheblichen Risikofaktor darstellen, da es von Massenspammern wie ein „offener Mailserver“ missbraucht werden könnte. Dadurch, dass das Script nun aber alle Mails an die angegebene Adresse sendet, wird es für
Spammer uninteressant.

Auf weitere Details im Perl-Code wollen wir nicht eingehen,

da Perl nicht Thema dieses Artikels ist. Die auskommentierten Zeilen im Code weisen darauf hin, was in den jeweils nachfolgenden Code-Abschnitten passiert. Die einleitend genannten Script-Aufgaben sind wiedererkennbar. Damit das Form-Mail-Script jedoch
funktioniert, müssen noch zwei wichtige Voraussetzungen erfüllt sein:

- Das CGI-Modul für Perl muss installiert sein. Seit Perl-Version 5.004 gehört es zu den Standardmodulen, die mit jeder Perl-Installation automatisch mit installiert werden. Da diese Perl-Version schon seit Jahren überholt ist, dürfte das CGI-Modul
  jedoch mit ziemlicher Sicherheit installiert sein. Im Zweifelsfall können Sie am Shell-Prompt eingeben: `perl -v`. Damit gibt der installierte Perl-Interpreter seine Versionsnummer aus.
- Das Programm **sendmail** muss installiert sein, und zwar unter /usr/lib/sendmail. Dieses Programm ist für das Versenden von Mails verantwortlich und wird innerhalb des Perl-Scripts aufgerufen. Geben Sie im Zweifelsfall am
  Shell-Prompt des Servers `which sendmail` ein, um zu überprüfen, ob und wo das Programm installiert ist. Bei Linux-Installationen gehört sendmail jedoch meistens zur Basis-Software.