

## WEB-Counter mit Perl

**Viele Homepages verkünden dem Betrachter stolz, wie oft sie schon abgerufen wurden. Meist handelt es sich bei diesen Countern um vorgefertigte Routinen die vom Provider oder einem anderen Dienstleister angeboten und in die eigene Homepage eingebunden werden. In diesem Artikel soll ein Blick hinter die Kulissen geworfen werden.**

### SSI – Server Side Includes

In den letzten ZPG-Mitteilungen befasste sich der Artikel „Wir basteln uns ein CGI-Script“ mit der dynamischen Erstellung von WEB-Seiten mittels CGI-Skripten in Perl. Oftmals sollen aber nur einzelne Teile einer WEB-Seite beim Aufruf dynamisch generiert werden, der Rest ist statisch. Hierzu wird in eine statische HTML-Seite in einem Kommentar versteckt ein einzelner Programmaufruf, z.B. `<!--#exec Befehl -->` eingebunden. Beim Abrufen der Datei, durchsucht der WEB-Server die zu übertragende WEB-Seite und ersetzt den Programmaufruf durch die Ausgabe des aufgerufenen Programms. Der Surfer bekommt nur das Ergebnis der Aktion übermittelt.

Dieses Verfahren wird als Server Side Includes, kurz SSI, bezeichnet. Um SSI beim Apache WEB-Server zu aktivieren sind einige Änderungen an der Konfigurationsdatei `httpd.conf` vorzunehmen. Unter SuSE Linux befindet sich die Konfigurationsdatei in `/etc/httpd/`. Als erstes muss der Kommentar vor den folgenden beiden Zeilen entfernt werden.

```
AddType text/html .shtml
AddHandler server-parsed
.shtml
```

Anschließend müssen Sie nach einer Zeile mit folgender Form suchen.

```
<Directory „/usr/local/httpd/htdocs“>
```

Die darauf folgenden Zeile, die mit *Options* beginnt, muss um die Option **Includes** erweitert werden.

```
z.B. Options Indexes FollowSymLinks Includes
```

Nach einem Neustart des WEB-Servers mit

```
/sbin/init.d/apache restart
```

werden alle *shtml Dateien* bei ihrem Abruf automatisch nach versteckten Programmaufrufen durchsucht (geparst). Normale *html Dateien* werden wie bisher direkt weitergeleitet.

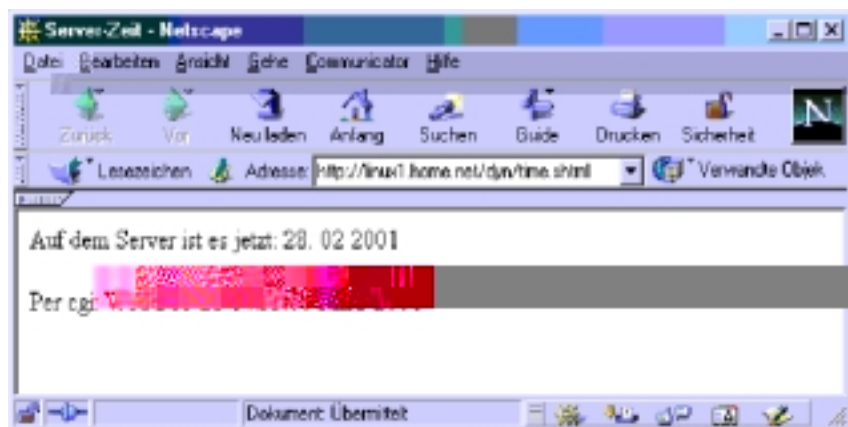
### Server-Zeit mittels SSI

Im ersten Beispiel soll innerhalb einer WEB-Seite die aktuelle Server-Zeit mit ausgegeben werden. Erstellen Sie dazu in Ihrem WEB-Verzeichnis (im Beispiel `/use/local/httpd/htdocs/dyn/`) die Datei `time.shtml` mit dem folgenden Inhalt.

```
<!-- /usr/local/httpd/htdocs/dyn/time.shtml-->
<html>
  <head>
    <title>Server-Zeit</title>
  </head>

  <body>
    Auf dem Server ist es jetzt: <!--#exec
    cmd="/bin/date +%d. %m %Y"-->
    <p>
      Per cgi: <!--#exec cgi="/cgi-bin/time.cgi"-->
    </body>
</html>
```

Der Aufruf von `time.shtml` führt zu folgendem Ergebnis, dass bei jedem Klick auf den *Neu laden Button* aktualisiert wird.



Die Ausgabe der Zeit erfolgt im Beispiel auf zweierlei Art. Im ersten Fall wird durch

```
<!--#exec cmd="/bin/date +%d. %m %Y"-->
```

auf dem Server der normale UNIX-Befehl `/bin/date` mit den angegebenen Parametern ausgeführt. Über `cmd="Befehl"` lassen sich demnach beliebige Unix-Befehle ausführen. Ein nicht zu unterschätzendes Sicherheitsrisiko!

Im zweiten Fall ermittelt das CGI-Script `time.cgi` die aktuelle Zeit.

```
<!--#exec cgi="/cgi-bin/time.cgi"-->
```

Dieses CGI-Script greift letztendlich ebenfalls auf den Unix-Befehl `/bin/date` zurück.

```
#!/usr/bin/perl
#
# /usr/local/httpd/cgi-bin/time.cgi
#
# Uebertraegt die Systemzeit des Servers
#
# Von: H. Baeurle - Sep. 2000

print „Content-type: text/html\n\n“;

chop ($zeit=`/bin/date`);
print $zeit;

# Programmende
```

Die Ansicht des Seitenquelltextes zeigt, dass der Browser lediglich das Ergebnis der Programme übermittelt bekommt, von den Programmaufrufen ist nichts mehr zu sehen.



## Text-basierter Zähler

Sollen nun die Anzahl der Aufrufe einer WEB-Seite erfasst werden, muss lediglich bei jedem Aufruf ein Programm gestartet werden, das einen Zähler ausliest, diesen um eins erhöht und anschließend ausgibt. Das nachfolgende CGI-Script in Perl erfüllt diese Aufgabe:

```
#!/usr/bin/perl
#
# /usr/local/httpd/cgi-bin/zaehler.cgi
#
# von H. Baeurle - letzte Aenderung:
# 27.02.2001

# Hier wird die Anzahl der Zugriffe
# gespeichert
$datei="/usr/local/httpd/htdocs/dyn/treffler";

# Zaehlerstand aus Datei holen
```

```
open(ZAEHLER,"<$datei");
$treffer=<ZAEHLER>;
close(ZAEHLER);

# Zaehlerstand um eins erhoehen und
# zurueckschreiben
open(ZAEHLER,">$datei");
print ZAEHLER ++$treffer;
close(ZAEHLER);

# Antwort an HTML-Seite ausgeben
print „Content-type: text/html\n\n“;
print „$treffer“;

# Programmende
```

Damit das Programm gestartet werden kann, müssen alle User über den Befehl `chmod o+x zaehler.cgi` das Recht zum Ausführen des Scriptes erhalten. Der aktuelle Zählerstand wird in der Datei `/usr/local/httpd/htdocs/dyn/treffler` abgelegt, die mit `touch treffler` in diesem Verzeichnis erstellt werden kann. Damit `zaehler.cgi` lesend und schreibend darauf zugreifen kann, müssen mit `chmod o+rw` noch die entsprechenden Zugriffsrechte gesetzt werden. Die folgende WEB-Seite setzt `counter.cgi` ein.

```
<!-- /usr/local/httpd/htdocs/dyn/
zaehler.shtml-->
<html>
  <head>
    <title>Ein einfacher Z&auml;hler</title>
  </head>

  <body>
    Das ist der <!--#exec cgi="/cgi-bin/
zaehler.cgi"-->. Aufruf.
    <p>
  </body>
</html>
```

Die nachfolgende Abbildung zeigt das Ergebnis der Aktion. Bei jedem Klick auf den *Neu-laden-Button* wird der Zähler um eins erhöht.



## Zähler mit graphischer Ausgabe

Nachdem die Funktion des Zählers überprüft wurde, pöppeln wir ihn optisch noch ein wenig auf. Dazu benötigen wir 10 Grafiken, die die Ziffern 0 bis 9 darstellen.



Diese Dateien *0.gif*, *1.gif* bis *9.gif* kopieren wir in das Verzeichnis, in dem auch die *html* Datei liegt. Die Anzahl der Treffer wird diesmal in der Datei */usr/local/httpd/htdocs/dyn/hints* gespeichert, die wieder mit *touch* erzeugt werden kann und mit den entsprechenden Zugriffsrechten versehen werden muss. Für das CGI-Script kopieren wir einfach *zaehler.cgi* nach *counter.cgi* und erweitern es wie folgt.

```
#!/usr/bin/perl
#
# /usr/local/httpd/cgi-bin/counter.cgi
#
# von H. Baeurle - letzte Aenderung:
27.02.2001

# Pfade
$bilderpfad="http://linux1.home.net/dyn/";
$datei="/usr/local/httpd/htdocs/dyn/hints";

#Zaehlerstand aus Datei holen
open(ZAEHLER,"<$datei");
$treffer=<ZAEHLER>;
close(ZAEHLER);

# Zaehlerstand um eins erhoehen und
zurueckschreiben
open(ZAEHLER,">$datei");
print ZAEHLER ++$treffer;
close(ZAEHLER);

#Antwort an HTML-Seite ausgeben
print "Content-type: text/html\n\n";

# Aus wieviel Stellen besteht die Zahl?
$stellen = split(// , $treffer);

# Stellen aufsplitten
@digits = split(// , $treffer);

# Bilder ausgeben
foreach $digit (@digits)
{
    print "<img
src=\"\$bilderpfad$digit.gif\">";
}

# Programmende
```

Für die Ausgabe wird mit *split()* die Zahl in *\$treffer* in ihre einzelnen Ziffern zerlegt und anschließend für jede dieser Ziffer in einer *for-Schleife* ein *Image-Tag* mit der zugehörigen Grafik ausgegeben. Das Ergebnis sieht dann wie folgt aus.



## Counter in der Praxis

Jeder der keinen eigenen WEB-Server betreibt wird aller Wahrscheinlichkeit auch keine Möglichkeit haben eigene CGI-Scripte bzw. SSI zu verwenden. Der Wunsch nach eigenen CGI-Scripten wird bei den meisten Providern auf taube Ohren stoßen – das Sicherheitsrisiko ist einfach zu groß.

In der Regel stellen die meisten Provider aber eine Reihe eigener CGI-Scripte zur Verfügung die der Kunde innerhalb seiner Homepage verwenden kann. Dazu gehört normalerweise auch ein Counter. Um den Zähler ohne SSI in die eigene Homepage zu packen, erfolgt der Script-Aufruf wie im folgenden Beispiel versteckt innerhalb eines *Image-Tags*.

```

```

Der Browser versucht nun die im Parameter *src* angegebene Grafik zu laden bei der es sich in diesem Fall aber um ein *CGI-Script* handelt. Das *CGI-Script* erzeugt aus den nach dem Fragezeichen übergebenen Parametern eine entsprechende Grafik. Im Beispiel wird der Zählerstand für die Homepage von User *maier* um eins erhöht und anschließend als sechsstellige Grafik ausgegeben. Falls der eigene Provider keinen Zähler anbietet, kann auch über die URL eines anderen Anbieters ein Zähler realisiert werden. Das *Image-Tag* hat dann den folgenden Aufbau.

```

```

Die in diesem Artikel aufgeführten Beispiele sind lediglich einfache Lösungsansätze. Bessere und wesentlich leistungsfähigere Scripte finden sich im Internet, z.B. unter *www.mueller-net.de* Rubrik *Homepage Zubehör*.

Harald Baeurle

E-Mail: [HaraldBaeurle@web.de](mailto:HaraldBaeurle@web.de)

•