Inhaltsverzeichnis

Arbeitsweise von serverseitigen Programmen

Der Namo Webeditor ist ein mächtiges Werkzeug bei der Erstellung von Webseiten. Ohne Probleme kann man schnell schöne Layouts entwickeln und auch beim Einfügen von Navigationselementen wird durch die Themenelemente einiges geleistet. Trotz allem darf man nicht vergessen, das man nur 'statische' Seiten erzeugt. Statisch bedeutet, dass sich der Inhalt der Seiten nach dem Hochladen auf den Server nicht mehr ändert. Auch Interaktionen mit dem Seitenbesucher sind nicht möglich.

Doch gerade das ist für die meisten Homepage Besitzer höchst interessant. Man möchte wissen, wie oft die Seiten aufgerufen wurden oder sogar ein Gästebuch in die Seite integrieren. Dazu muss man zunächst wissen, wie die Kommunikation zwischen dem Surfer (Client) und dem Server funktioniert.

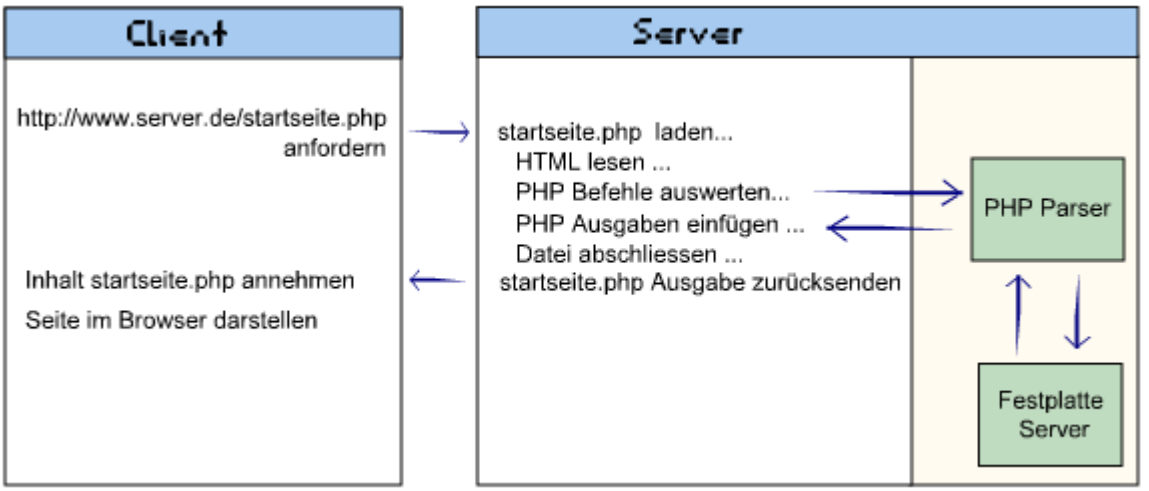
Der Client schickt eine URL an den Server. Z.B http://www.server.de/startseite.html . Der Server erkennt an der Dateiendung .html, das es sich um eine HTML-Seite handelt, liest den Inhalt der Datei und schickt sie dem Client. Die Auswertung (und Darstellung) einer HTML-Datei ist allein Sache des Browsers und kümmert den Server deshalb nicht.

Um dem Server nun mitzuteilen, das er die Inhalte der Dateien auch parsen (auswerten) soll, ist ihm das über die Dateiendung mitzuteilen. Die Dateiendung .php weist den Server an nicht nur evtl. vorhandenen HTML-Quelltext auszugeben, sondern die Datei nach PHP-Befehlen zu durchzusuchen und diese auch auszuführen.

Der Vorteil von PHP (der auch zu seiner weiten Verbreitung geführt hat) liegt in der Tatsache, das man HTML- und PHP-Quelltext beliebig mischen kann. Da PHP eine 'richtige' Programmiersprache ist, kann man innerhalb von HTML Programme schreiben, die Dateien auf dem Server anlegen, öffnen, schreiben, verändern oder Datenbankanfragen absetzen oder, oder, oder.

Mit Pearl ist das nicht ganz so einfach, da man den Pearl Code nicht so ohne weiteres mit HTML mischen kann. Das geht dann nur mittels SSI (Server Side Includes). Pearl Scripte liegen als programmierter Text immer alleine als Datei auf dem Server und werden dort von der Pearl.exe oder als Apache-Modul (dann meist .cgi) ausgeführt. Die HTML Ausgabe muss dann komplett von Pearl erzeugt werden. Das ist relativ aufwändig und lässt Designtools außen vor. Nur das Arbeiten mit Templates bietet hier noch ausreichend Möglichkeiten.

Den ganzen Prozess kann man so veranschaulichen:



Ein sehr einfacher Counter mit PHP eingebettet in HTML könnte so aussehen:

startseite.php

```
<html><head>
<title>PHP Beispiel Script</title>
</head>
<body>
<?php         // mit diesem tag beginnt die php ausführung
$counter = file("counter.txt");
              // liesst die ganze datei 'counter.php' in das array '$counter'.
              // jede zeile 1 element
$counter = $counter[0] + 1;
              // nimmt das 1. element des arrays, addiert 1
              // und speichert es als neuen inhalt in die variable zurück
$fp = fopen("counter.txt", "w");
              // öffnet 'counter.php' zum schreiben.
              // $fp ist das dateihandle (zeiger)
fwrite($fp, $counter);
              // schreibt den inhalt der variablen $counter
              // in die datei mit dem handle $fp
fclose($fp);  // schliesst die datei
?>            // php ausführung beenden
<P>Dies ist HTML Ausgabe</P>
<P> Diese Seite wurde <?php echo $counter; ?> mal aufgerufen.</P>
              // 'echo' schreibt auf den Bildschirm
              // den inhalt der variablen $counter
              // der variableninhalt von $counter ändert sich erst nach erneutem
              // aufruf der seite und steht so lange in PHP zur verfügung
</body>
</html>
```

Diesen Counter in Perl zu implementieren erfordert 2 Dateien auf dem Server und SSI Unterstützung. Zunächst die Pearl-Programmdatei:

counter.pl

```
open (FP, "<counter.txt");
          // öffnet counter.txt lesend (<) und schreibt das dateihandle in FP
$counter = <FP>;
          // in die variable $counter wird der inhalt der datei
          // mit dem handle FP geschrieben
close (FP);
          // die datei wird geschlossen
$counter = $counter + 1;
          // der inhalt von $counter (sollte eine zahl sein ;-) ) wird um 1 erhöht
          // und zurück in die variable geschrieben
open (FP, ">counter.txt");
          // counter.txt wird zum schreiben (>) geöffnet - dateihandle in FP
print FP $counter;
          // der inhalt der variablen $counter wird in die datei
          // mit dem handle FP geschrieben
close (FP);
          // counter.txt wird geschlossen
print "Content-type: text/html";
          // dem server mitteilen, was für eine art von daten nun kommt
print $counter;
          // das ist die anzahl der aufrufe
```

Nun die dazugehörige HTML-Datei:

startseite.xxx

```
<html><head><title>PHP Beispiel Script</title></head>
<body>
<p>Dies ist HTML Ausgabe</p>
<p> Diese Seite wurde
<!-- #exec cgi="/counter.pl" -->
 mal aufgerufen.</p>
</body>
</html>
```

Das als Dateiendung hier .xxx steht ist nicht ohne Grund. Auch in diesem Fall muss dem Server über die Dateiendung mitgeteilt werden, mit welcher Erweiterung/Programm die Datei geparst werden muss. Ansonsten wird die **<!-- #exec cgi ... -->** Zeile einfach übersehen!
Eine allgemeingültige Dateiendung gibt es dafür nicht. Sie ist individuell festlegbar. Der Provider wird diese Information zur Verfügung stellen.

Ohne SSI Unterstützung muss das Beispiel in Perl in einer einzigen Perl-Datei realisiert werden und der Aufruf der Datei geschieht direkt über die URL:

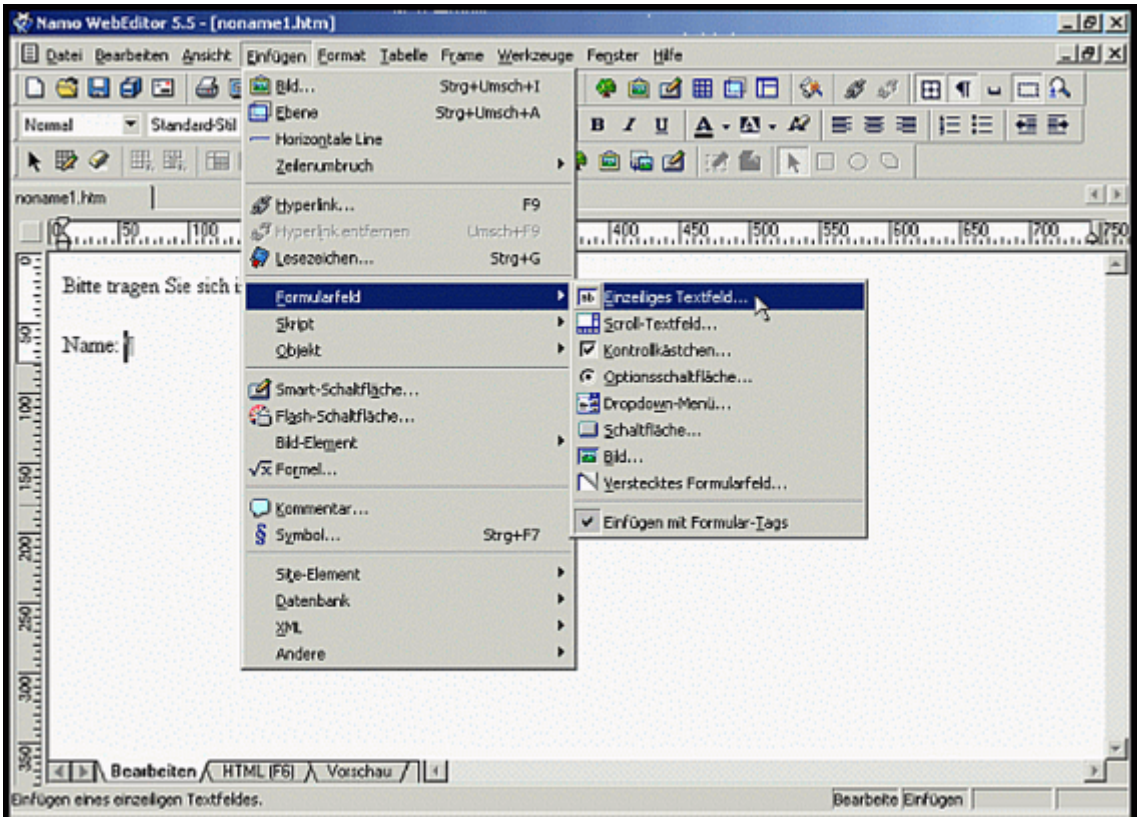
startseite.pl / startseite.cgi

```
open (FP, "<counter.txt");
$counter = <FP>;
close (FP);
$counter = $counter + 1;
open (FP, ">counter.txt");
print FP $counter;
close (FP);
print "Content-type: text/html";
print "<html><head><title>PHP Beispiel Script</title></head>";
print "<body>";
print "<p>Dies ist HTML Ausgabe</p>";
print "<p> Diese Seite wurde ";
print $counter;
print "  mal aufgerufen.</p>";
print "</body> </html>";
```

Wie man sieht, ist diese Art der Website Programmierung mit Perl nicht sehr komfortabel und schließt WYSIWIG Editoren wie Namo, Dreamweaver und andere aus.
Das ist mit PHP anders. Das Editieren ist dort mit Namo problemlos möglich. An den Stellen, an denen PHP-Scripte eingefügt wurden, erscheint lediglich ein kleines PHP Symbol.

Bei exzessiver Verwendung von manuell eingebrachtem PHP-Code sollte man aber ständig ein Auge auf den Quelltext haben. Namo korrigiert solche Scripte manchmal kaputt. So ist es mir schon des öfteren passiert, das Namo <?php in <?php geändert hat. Damit erkennt das Parser auf dem Server natürlich nicht mehr den Beginn der PHP-Befehle und gibt die Befehlsfolgen als Text an den Client zurück.

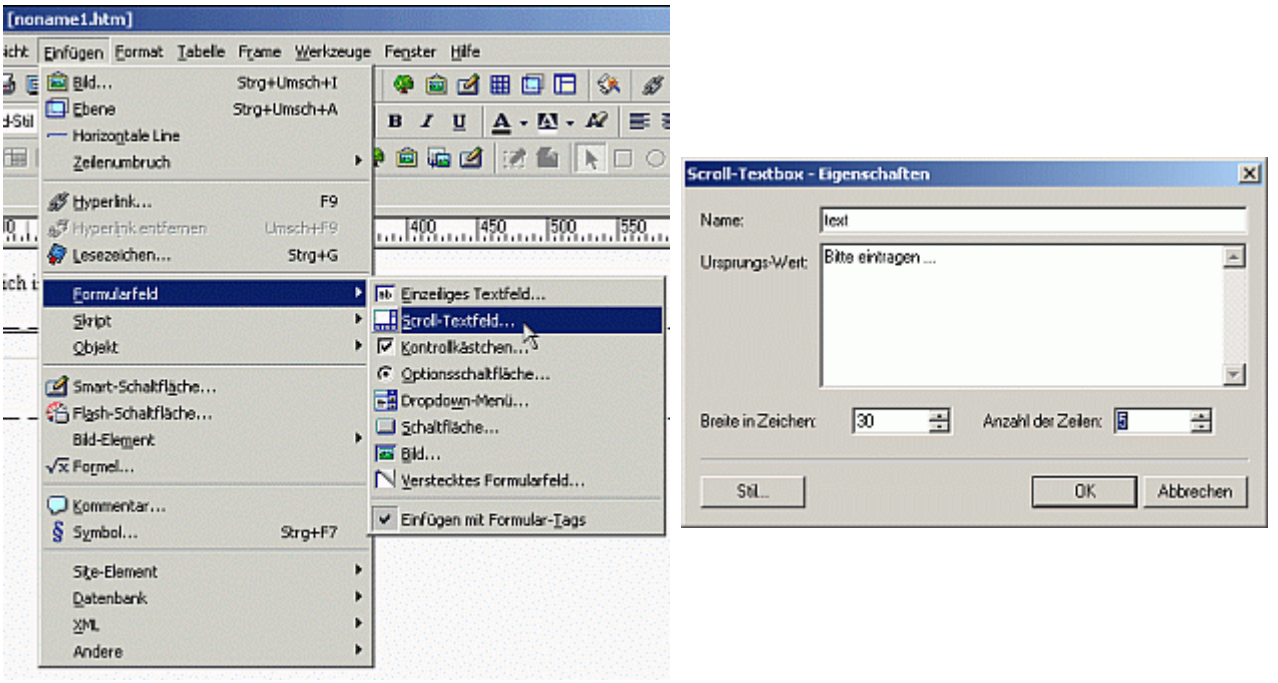
Das Prinzip des Counters läßt sich natürlich auch auf Text anwenden. Zur Verdeutlichung soll nun ein einfaches Gästebuch mit PHP unter Verwendung von Namo 5.5 entwickelt werden.
Zunächst wird die Eingabemaske mit Namo erstellt. Dazu öffnet man eine leere Seite, gestaltet sie und setzt den Cursor an die Position, an der der Besuchername eingegeben werden soll. Nun fügt man wie hier zu sehen ein einzeiliges Textfeld ein.



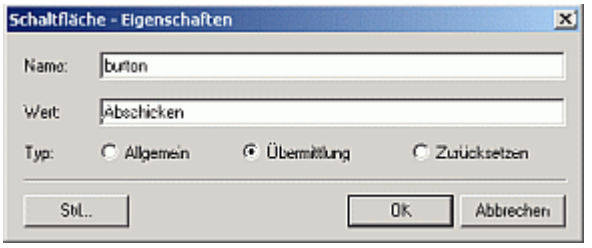
In der nun öffnenden Eigenschaftsbox sind folgende Eingaben zu machen.

Dadurch legt man den Namen der Übergabevariablen auf 'name' fest, die optische Breite der Eingabezeile auf 30 und die maximale Länge des Eingabetextes wird auf 50 Zeichen begrenzt.

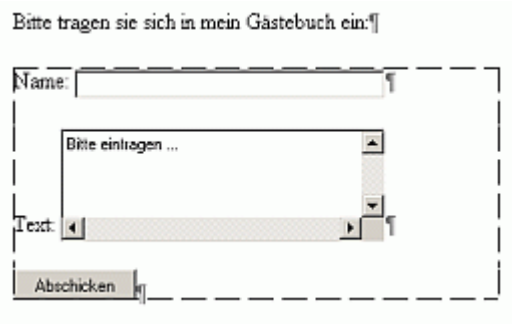
Nun fügt man ein mehrzeiliges Textfeld für den Kommentar ein und legt die Eigenschaften fest.



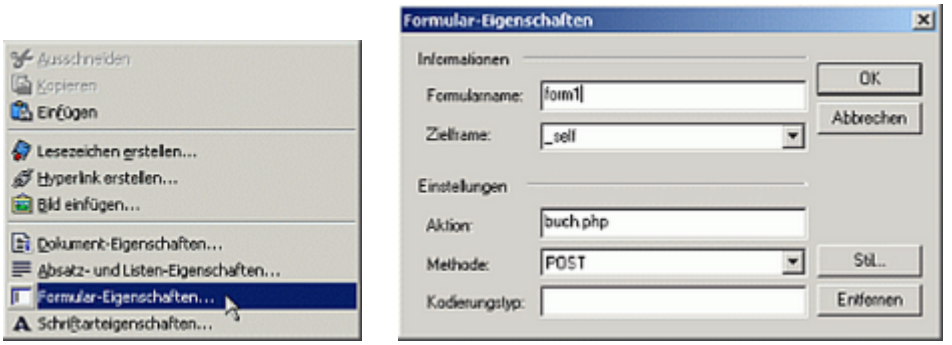
Zum Schluss wird noch ein Button zum Absenden eingefügt. Mit dem Wert legt man die Beschriftung des Knopfes fest. Im Optionsfeld muss 'Übermittlung' angeklickt sein.



Bitte achten Sie darauf, das die Felder gemeinsam in einem Form Bereich liegen (gestrichelte Linie), da sie zusammen gehören und die Inhalte zusammen an das Folgescript übergeben werden müssen.



Durch einen Rechtsklick in den Form Bereich und Wahl der 'Formular Eigenschaften ...' legt man jetzt die Aktion (das Empfängerscript) des Formulars fest. Der Formname ist eigentlich nicht wichtig. Er dient nur zur Unterscheidung bei mehreren eigenständigen Formularen auf einer Seite. Als Zielframe legen Sie am besten '_self' fest. Dorthin werden die Folgeausgaben gemacht. Unter Aktion gibt man nun die Adresse des Scriptes an, daß die Inhalte entgegennimmt und weiterverarbeitet. Als Methode wählt man POST. Der Kodierungstyp ist der MIME-Typ für den Formularinhalt. Ist der Inhalt gewöhnlicher Text, so kann dieses Feld frei gelassen weden.



Dieser Quelltext wird dabei erzeugt:

gaestebuch.html

```
<html><head><title>Gästebuch</title></head>
<body bgcolor="white" text="black" link="blue" vlink="purple" alink="red">
<p>Bitte tragen Sie sich in mein Gästebuch ein:</p>
<form name="form1" target="_self" method="post" action="buch.php">
<p>Name: <input type="text" name="name" maxlength="50" size="30"></p>
<p>Text: <textarea name="text" rows="5" cols="30">Bitte tragen Sie sich
ein ...</textarea></p>
<p><input type="submit" name="button" value="Abschicken"></p>
</form>
</body></html>
```

Als nächstes wird das PHP Script geschrieben. Dieses Script kann nun ohne vorherige Definition auf die gesendeten Variablennamen ('name' und 'text') zugreifen (Variablennamen werden in PHP ein '$' vorangestellt).

buch.php

```
<?php
$eintrag = $name . "|" . $text . " ";
$fp = fopen("gaestebuch.txt", "a+");
fwrite($fp, $eintrag);
fclose($fp);
?>
Zeile 2:
$eintrag -> zusammengesetzter eintrag
$name    -> Name des Schreibers
. "|"    -> Trennzeichen zwischen Name und Kommentar einfügen
. $text  -> Kommentar hinzufügen
. " "   -> Zeilenvorschub hinzufügen
Zeile 3:
"a+"     -> (a) öffnet datei zum schreiben und setzt filepointer ans ende
         -> (+) wenn datei nicht vorhanden wird sie erstellt
```

Allerdings kann das Script nicht so bleiben. Durch die Angabe von '_self' im Sendeformular wird der Inhalt des Fensters von diesem Script überschrieben. Nur - da ist nichts was ausgegeben wird. Das ändern wir jetzt, indem wir das Script in HTML einbetten.

buch.php

```
<html><head>
<title>Gästebuch</title></head>
<body>
<?php
$eintrag = $name . "|" . $text . "\n";
/* erweiterung */
$fp = fopen("gaestebuch.txt", "a+");
fwrite($fp, $eintrag);
fclose($fp);
?>
<p>Vielen Dank für Ihren Eintrag.</p>
</body>
</html>
```

So funktioniert das Gästebuch bereits. Jeder Eintrag kommt in eine Zeile. Jetzt fehlt noch ein Script für die Ausgabe. Etwas sehr einfaches ginge so:

anzeigen.php

```
<html>
<head><title>Gästebuch</title></head>
<body>
<p>Die Gästebucheinträge</p><hr>
<p>
<?php
$fd = fopen ("gaestebuch.txt", "rb");
$buch = fread ($fd, filesize ("gaestebuch.txt"));
fclose ($fd);
$buch = str_replace("\n ","</p>\n <p>",$buch);
$buch = str_replace("|","<br>",$buch);
echo $buch;
?>
</p>
</body>
</html>
```

Bei der Ausgabe wird hier zunächst ein Absatz geöffnet ( ). Dann wird das Gästebuch gelesen und die Zeilenvorschübe ("\n ") durch Absatzende und Absatzanfang (" \n ") ersetzt. Dadurch erhält jeder Eintrag einen eigenen Absatz. Damit Name und Kommentar nicht nebeneinander stehen, wird das Trennungszeichen ("|") durch einen Break ("
") ersetzt.

Im Prinzip ist das Gästebuch damit voll funktionsfähig. Aber hier ist noch ein Wort zur Vorsicht angebracht. Es ist bei diesem Gästebuch möglich auch PHP oder Javascript einzugeben. Diese Scripte würden alle ausgeführt werden und stellen ein hohes Sicherheitsrisiko dar.
Um das zu verhindern fügt man sicherheitshalber noch folgende Zeilen bei "**/*erweiterung*/**" ein:

```
$name = strip_tags($name);    oder   $name = htmlentities($name);
$text = strip_tags($text);    oder   $text = htmlentities($text);
```

Dadurch werden alle HTML, Javascript und PHP Tags vor der Weiterverarbeitung entfernt oder in HTML-Zeichen umgewandelt.

Schön ist das Gästebuch nicht und auch nicht sehr felxibel. Das Eingabeformular setzt man am besten in eine Tabelle, um es besser formatieren zu können.
An der Ausgabe muss auch noch einiges gemacht werden. Die einzelnen Eingaben (Zeilen) müsste man sauber trennen, damit man über eine Schleife gezielt eine bestimmte Anzahl von Einträgen pro Seite ausgeben kann.
Hilfreich dabei sind die PHP-Funktionen **explode()**, **implode()**, **count()** und **for{}**

All das - und mehr - sein dem experimentierfreudigem Webdesigner überlassen .... ;-)

E-mail Versand

Serverunterstützung -und erlaubnis vorrausgesetzt, ist es auch möglich, die Daten des Eingabeformulars per

e-mail zu versenden.

Dieses Script zeigt kurz, welche PHP-Funktionen dafür einzusetzen sind.

```
<html>
<head>
<title>e-mail Versand</title>
</head>
<body>
<?php
$eintrag = "Name: " . $name . "\n " . "Text: " . $text;
$empfaenger = "webmaster@namouser.de";                // dort geht die e-mail hin
$subjekt = "e-mail Anfrage Webmaster NamoUser.de";    // betreffzeile
mail($empfaenger, $subject, $eintrag);
?>
<p>Ihre Nachricht wurde an den Webmaster gesendet.</p>
</body>
</html>
```

Ich hoffe, das ich mit diesem Artikel einige Unklarheiten beseitigen konnte. Sollten sich Fehler eingeschlichen haben bitte ich um eine **kurze Nachricht**.

[wf]