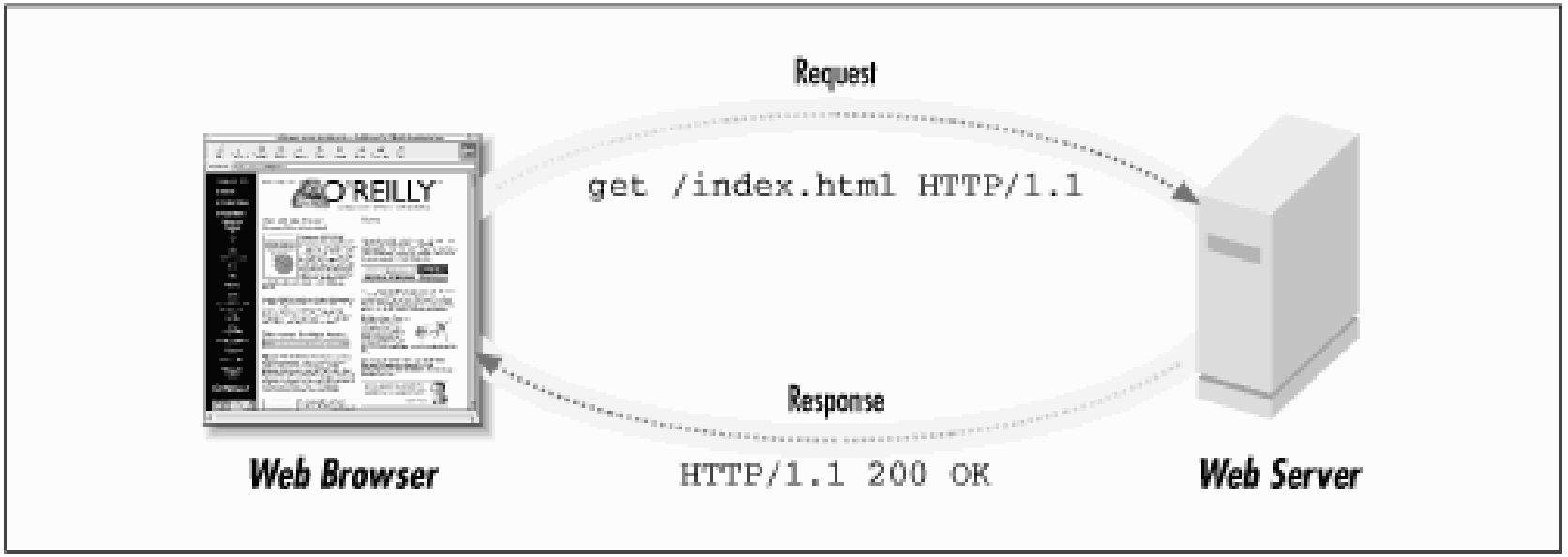
Apache Module
Funktion, Konfiguration und Programmierung

- Abruf einer Webseite, Request/ Response Cyle
- Einbindung von Modulen in Apache
- Einsatz von Standardmodulen
- Programmierung von Modulen

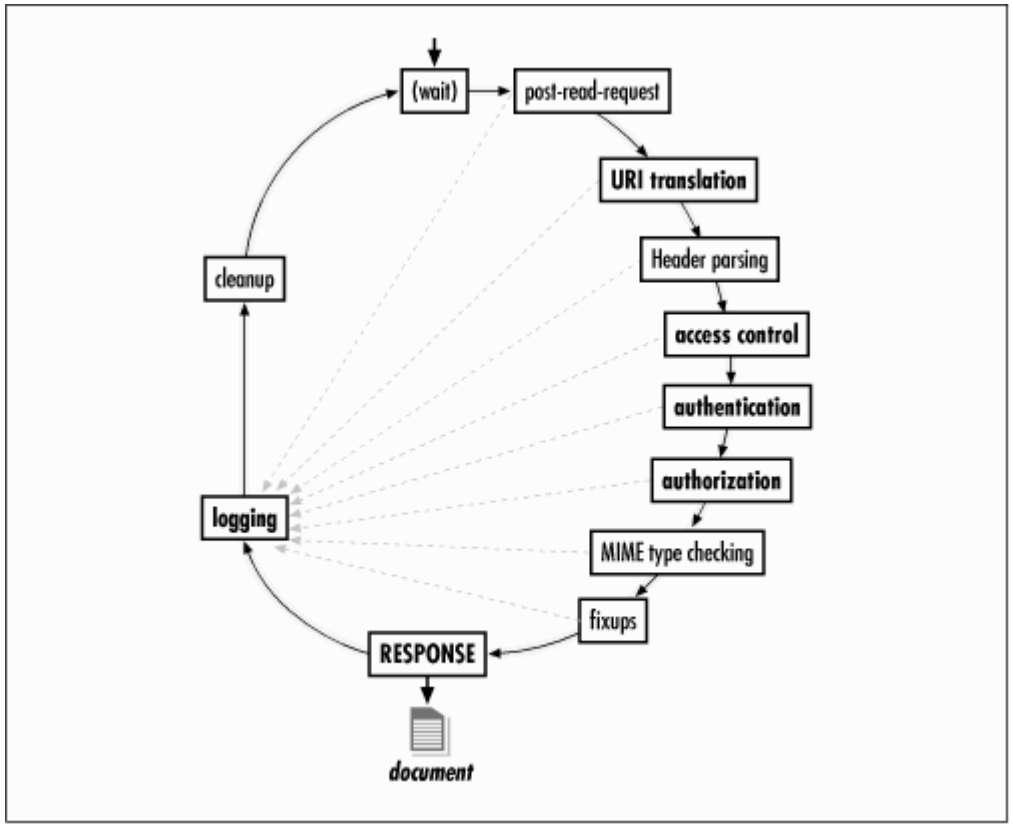
Request/ Response Cycle

- Client (Browser) fordert eine Seite vom Webserver an
- Webserver bearbeitet Anfrage und liefert eine Seite aus
- Austausch von Header Informationen über http



Einzelne Phasen des Requests

- Abbildung der URI auf Dateisystem
- Zugriffskontrolle
- Antwort
- Logging



Module

Für jede Phase des Request können Module definiert und in Apache integriert werden.

- Konfiguration in httpd.conf
- API für C und Perl (mod_perl)
- Handler (Funktion) wird von Apache aufgerufen
  - Dynamische URL- Umwandlung, Aufruf des Folgehandlers
  - Request mit entsprechendem Returncode abbrechen
  - Header ändern
  - Subrequests auslösen
  - Inhalt erzeugen

Standardmodul mod_alias

httpd.conf

```
Alias /image /ftp/pub/image
```

Beim Aufruf von `http://myserver/image/foo.gif` liefert der Server die Datei `/ftp/pub/image/foo.gif` aus.

httpd.conf

```
Redirect /service http://foo2.bar.com/service
```

Der Aufruf von `http://myserver/service/foo.txt` wird nach `http://foo2.bar.com/service/foo.txt` umgeleitet.

Problem

- Webserver wird zur Dokumentenablage verwendet
- Im Dateinamen sind ID, Datum und Titel kodiert
- Bei Änderungen ändert sich das Datum und damit der Dateiname und Links werden ungültig
- Die ID bleibt gültig

```
ls -lR
.:
insgesamt 12
drwxr-xr-x  2 hu hu 4096 2006-04-30 20:40 dir1
drwxr-xr-x  2 hu hu 4096 2006-04-30 20:42 dir2
-rw-r--r--  1 hu hu 1523 2006-04-30 20:51 redirect.pl

./dir1:
insgesamt 16
-rw-r--r--  1 hu hu 122 2006-04-30 20:34 doc_03_20060213_Das ist ein Titel.html
-rw-r--r--  1 hu hu 122 2006-04-30 20:35 doc_05_20060313_Das ist noch ein Titel.html
-rw-r--r--  1 hu hu 126 2006-04-30 20:38 doc_0815_20060424_Kontrabass und Bratsche.html
-rw-r--r--  1 hu hu 128 2006-04-30 20:40 doc_22356_20060501_Schwarz und nicht schwarz.html

./dir2:
insgesamt 16
-rw-r--r--  1 hu hu 122 2006-04-30 20:36 doc_09_20060323_Wieder ein ein Titel.html
-rw-r--r--  1 hu hu 122 2006-04-30 20:37 doc_11_20060307_Extremklettern leichtgemacht.html
-rw-r--r--  1 hu hu 127 2006-04-30 20:41 doc_1724_20060401_titel.html
-rw-r--r--  1 hu hu 126 2006-04-30 20:39 doc_4711_20060414_Ordnung und Sauberkeit.html
```

Standardmodul mod_rewrite

Konfiguration in .htaccess

```
Options +FollowSymLinks +Indexes
# Dateinamen der Form doc_[Zahl].html werden an redirect.pl übergeben. In %{SCRIPT_FILENAME}
# steht der absolute Pfad.
# redirect.pl macht ein ls und liefert ein Redirect auf den richtigen Namen
RewriteEngine On
RewriteCond  %{REQUEST_FILENAME} .*doc_[0123456789]+\.html
RewriteRule  ^(.*)$        redirect.pl?full=%{SCRIPT_FILENAME}  [L]
```

redirect.pl

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

use strict;
use CGI qw/:standard /;
use URI::Escape;
use File::Basename;
my ($fh,$line);
my $full = param("full");
my $cmd = "/bin/ls -1 '".substr($full,0,length($full)-5)."'".'*';
open($fh,"$cmd |");
$line = <$fh>;                                  # Erste Datei nehmen, die ls liefert
close($fh);
chomp($line);                                   # LF am Ende wegmachen
if( $line eq "" ) {                             # Keine passende Datei gefunden
    $line = substr($full,0,length($full)-5);    # .html am Ende entfernen
}                                               # Die Datei wird dann natürlich nicht gefunden

$line = fileparse($line);                       # Verzeichnisname entfernen. Dateiname bleibt
$line = uri_escape($line);                      # Dateiname maskieren (# in %23 umwandeln, usw.)

print redirect($line);                          # REDIRECT Header mit richtigem Namen schicken
```

Response

Konfiguration

```
PerlModule Module
<Location /mod>
    SetHandler perl-script
    PerlResponseHandler  Module
</Location>
```

Modul

```
package Module;

use strict;

use Apache2::RequestRec ();
use Apache2::Const -compile => qw(OK);
use CGI qw(:standard);

sub handler {
    my $r = shift;

    $r->content_type('text/html');                  # Header ausgeben
    $r->err_headers_out->add('Foo' => 'Bar');

    print start_html,"\n";                          # HTML-Code ausgeben
    print h1("Titel");
    print "Text der Seite\n";
    print end_html,"\n";

    Apache2::Const::OK;
}
1;
```

Authentication

Konfiguration

```
PerlModule AuthAny
<Location /authany>
   AuthName Test
   AuthType Basic
   PerlAuthenHandler AuthAny
   require valid-user
</Location>
```

Modul

```
package AuthAny;

use strict;
use Apache2::Const qw(:common);

sub handler {
    my $r = shift;
    my($res, $sent_pw) = $r->get_basic_auth_pw;
    return $res if $res != OK;

    my $user = $r->user();
    if( $user ne $sent_pw ) {
        $r->note_basic_auth_failure;
        $r->log_reason("username und password sind nicht gleich ", $r->filename);
        return AUTH_REQUIRED;
    }
    return OK;
}
1;
```

Error Handling

Im Fehlerfall wird ein Subrequest auf den Errorhandler ausgelöst.
Der Errorhander hat Zugriff auf alle Informationen des ursprünglichen Requests.

Konfiguration in httpd.conf

```
PerlModule error
<Location /error>
    SetHandler perl-script
    PerlResponseHandler  error
</Location>

ErrorDocument 404 /error?code=404
```

Konfiguration in .htaccess

```
PerlSetVar Webmaster "Hans-Jürgen Husel"
PerlSetVar MSG_404 "Keine Ahnung wo die ist"
```

Error Handling

Modul

```
package error;
use ...
sub handler {
    my $r = shift;
    my $webmaster = $r->prev()->dir_config('WEBMASTER');
    my $msg_404 = $r->prev()->dir_config('MSG_404');
    my ($code,$str,$title);
    $code = param("code") || ($code = 0);
    if( $code == 404 ) {                    # Not Found
        $title = "Seite nicht gefunden";
        $str = h1("Seite nicht gefunden")."Die von ihnen angeforderte Seite ".
           p(b($r->prev()->uri()))."gibt es nicht.\n";
        $str .= p("Bei Fragen wenden sie sich bitte an $webmaster.");
        }
        else {
            $str .= p("Kein Webmaster angegeben");
        }
        if( defined($msg_404) ) {
            $str .= p($msg_404);
        }
    }
    $r->content_type('text/html');
    print start_html($title),"\n";
    print start_table({-border => 0, -cellpadding => 3, -width => '100%' });
    print Tr(td({-valign=>"top", -bgcolor=>"#FFCCCC"},$str));
    print Tr(td(img({-src=>"/apache2-default/apache_pb.gif"})));
    print end_table();
    Apache2::Const::OK;
}
1;
```

Filter

Filter wird am Ende des Request vor der Auslieferung angewendet.
Wird auch auf das Ergebnis von dynamischen Seiten angewendet.
Beispiel: Einheitliche Fußnoten

Konfiguration

```
PerlModule Footer

<Location /filter>
    PerlOutputFilterHandler Footer
</Location>
```

Filter

Modul

```
package Footer;

use ...
use base qw(Apache2::Filter);
use constant BUFF_LEN => 1024;
use constant OVERLAP => 32;

sub handler : FilterRequestHandler {
    my $f = shift;
    my $r = $f->r();
    my $buffer = "";
    my $snippet = "";
    my(@arr,$i);
    my $footer = '<hr><a href="/info?uri='.$r->uri().'">Info<a/>'."\n";
    $r->headers_out->unset('Content-Length');
    while ($f->read($buffer, BUFF_LEN)) {
        @arr = split(/\n/,$buffer);
        $arr[0] = $snippet.$arr[0];
        for( $i=0 ; $i<$#arr ; $i++ ) {
            $arr[$i] =~ s!(</BODY>)!$footer $1!gi;
            $f->print($arr[$i]."\n");
        }
        $snippet = $arr[$#arr];
    }
    $snippet =~ s!(</BODY>)!$footer$1!goi;
    $f->print($snippet."\n");
    Apache2::Const::OK;
}
1;
```

Literatur

- Livehttpheaders für Firefox
  http://livehttpheaders.mozdev.org/
- Apache Dokumentation
  http://httpd.apache.org/docs/2.0/
- mod_perl Dokumentation
  http://perl.apache.org/docs/index.html
- Writing Apache Modules with Perl and C
  http://www.modperl.com/