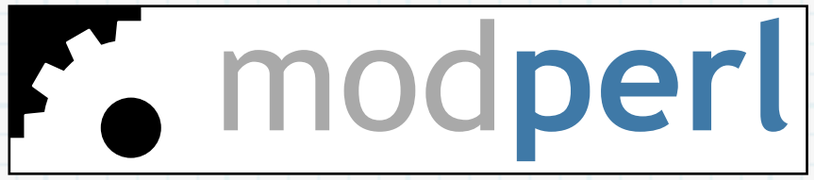

Entwicklung von Apache-Modulen in Perl

oder: wie kann ich einen Werwolf aus meinem Webserver aussperren, ohne eine vorhandene Applikation anpassen zu müssen.

Stuttgarter Webmontag, 31. Juli 2006

Rolf Schaufelberger, Alvar Freude

Um was geht es denn hier?

* Techniken für Webentwickler / Programmierer
* Verbindung einer mächtgen Script-Sprache mit einem Web-Server:

  Apache
  + Perl
  = mod_perl

mod_perl ist nicht nur ein Inhaltslieferant

* Man kann mit mod_perl ganz normal Web-Anwendungen Programmieren
* Oder eines der vorhandenen Frameworks nutzen
* Wir haben aber gleichzeitig den vollen Zugriff auf alle Internas des Apache-Webservers
* sprich: man kann beliebige Apache-Module auch in Perl schreiben

Es lassen sich alle Request-Phasen des Apache nutzen

* Authentifizierung
* Zugriffskontrolle
* Inhalte
* Filter
* URL-Umschreibung
* usw ...

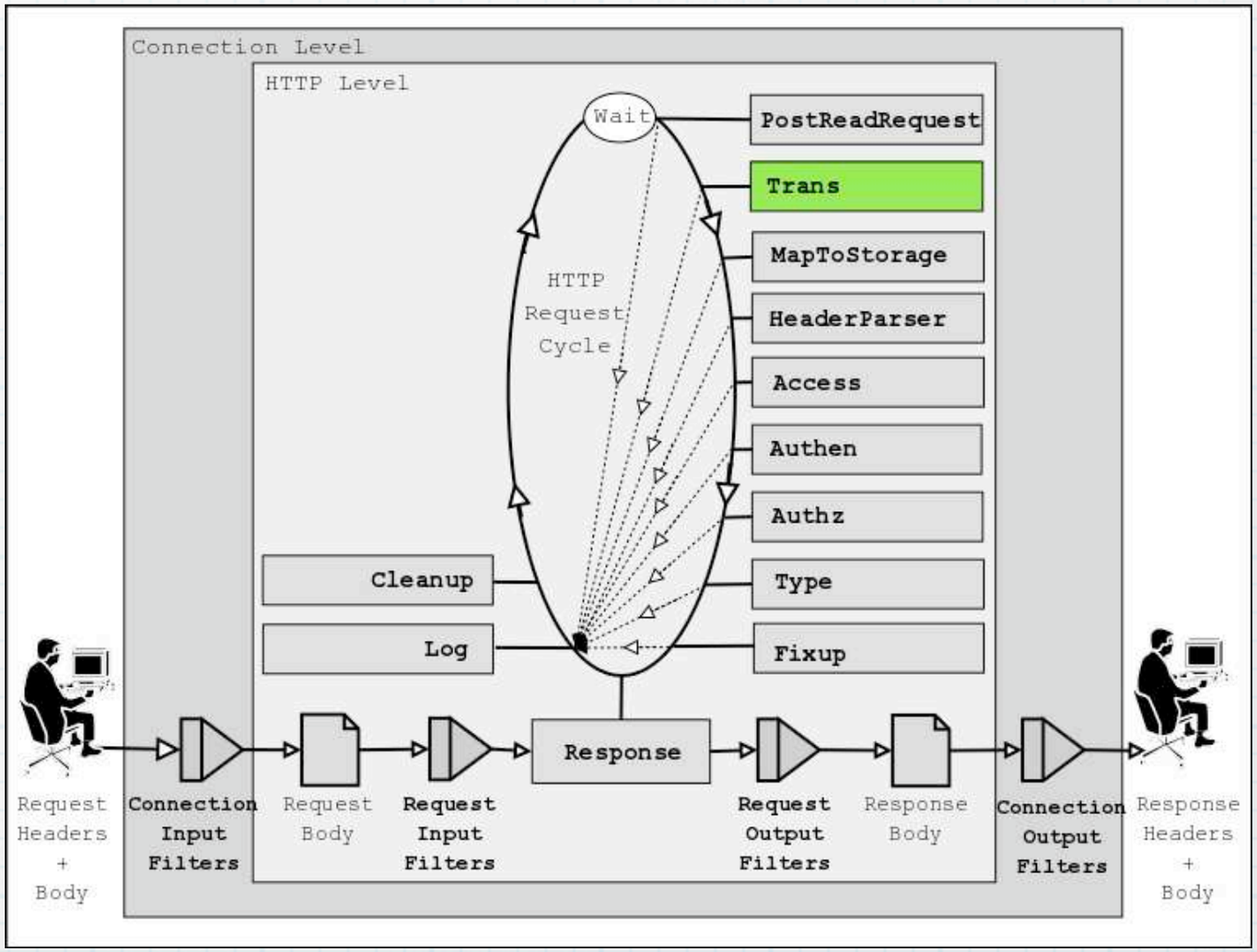

Diagramm ist schamlos kopiert von Stas Bekman: http://www.stason.org/

Beispiel

* Wir haben einen Apache-Webserver und eine fertige Applikation
* Diese soll vor Angriffen von Werwölfen geschützt werden
* Wir wollen an der Applikation nichts ändern
* Diese kann in jeder beliebigen Sprache geschrieben sein, Perl, PHP, Java ... – egal!

Die Lösung

* Der Apache arbeitet einen Zugriff in mehreren Phasen ab; jeder Phase können Handler zugewiesen werden
  * z.B. Translation (z.B. mod_rewrite), Access , Authen, Response, Log, ...
  * Wir brauchen den Access-Handler, denn da wollen wir den Werwolf aussperren

```
package Apache2::Werewolf;
use Astro::MoonPhase;

[...]

sub handler {
  my $hour = (localtime)[2];
  return FORBIDDEN if phase() > 0.9
          and ($hour >= 22 or $hour < 6);
  return OK;
}
```

Das ist unser Code

```
PerlModule Apache2::Werwolf

<Location /werwolf-sicheres/verzeichnis>
  PerlAccessHandler Apache2::Werwolf
</Location
```

Das steht in der Apache-Konfiguration

Weitere Beispiele: Trans-Handler

* Problem: Relaunch einer Website, alle URLs ändern sich

* Lösung: man schreibe einen PerlTransHandler, der nimmt anhand einer internen Tabelle (als Hash) oder mit Hilfe einer Datenbank eine Umsetzung vor

Noch ein Beispiel: Apache-Konfiguration

* Auch die gesamte Konfiguration des Apache-Servers oder einzelner virtueller Hosts lässt sich mit mod_perl realisieren
* Das kann hilfreich sein, wenn man viele virtuelle Hosts hat und diese mehr oder minder automatisiert konfiguriert werden sollen

```
<Perl>
  $VirtualHost{"www.foo.com"} =
    {
    DocumentRoot => "/tmp/docs",
    ErrorLog     => "/dev/null",
    Location     =>
        {
        "/" => {
                Allowoverride => 'All',
                Order         => 'deny,allow',
                Deny          => 'from all',
                Allow         => 'from foo.com',
                },
        },
    };
</Perl>
```

Vorteile gegenüber Modulen in C

* Schnellere Entwicklungszeit (Script-Sprache!) und API-Vereinfachungen (z.B. bei Filtern)
* Bessere Wartbarkeit und Pflege
* Viele fertige Perl-Module http://search.cpan.org/
* Hohe Plattformunabhängigkeit
* Keine Sicherheitslücken durch Buffer-Overflows

Response-Handler

* Der Response-Handler ist zum Ausliefern des Inhaltes einer Website da
* Kennen wir von PHP, CGI, statischen Seiten, Java usw.
* Das geht natürlich auch in mod_perl (häufig genutzt als einfacher CGI-Beschleuniger)
* Standard-Module und ganze Frameworks übernehmen viele Alltagsaufgaben

Frameworks

* Es gibt eine Reihe von Frameworks, die mod_perl als Backend nutzen
  * Mason
    Templating-System mit OO-Features und vielem mehr, u.a. Amazon
  * Catalyst
    MVC-Web-Framework, quasi Ruby-on-Rails mit Perl+CPAN
  * Embperl, Apache Axkit (XML-Server), u.v.m.

Perl

mod_perl bietet natürlich alle Features, Vorteile und Besonderheiten von Perl:

* Compilierte Script-Sprache (Bytecode)
* Namensräume
* Ties und Operator-Overloading
* über 10 000 Module auf dem CPAN verfügbar
* use strict; use warnings;
* komplettes OO inkl. Mehrfachvererbung
* Tolle Testing-Tools
* Sicherheits-Funktionen
* Ausführliche Dokumentation
* komfortable Datenbank-Anbindung

Links

* Dieser Vortrag:
  http://alvar.a-blast.org/vortraege/webmontag/mod_perl/
* http://perl.apache.org/
* Vorträge vom mod_perl-Chef-Entwickler
  http://www.stason.org/talks/
* Perl allgemein
  http://www.perl.org/ http://www.perl.com/
* Perl-Module (allgemein)
  http://search.cpan.org/

Danke, das wars!
Fragen?

modperl

Dipl-Ing. Rolf Schaufelberger

http://www.plusw.de/

rs@plusw.de

Dipl-Designer (FH) Alvar Freude

http://alvar.a-blast.org/

alvar@a-blast.org

Perl mongers

http://stuttgart.pm.org/

Perl

Und im nächsten Vortrag geht es um eine Anwendung von mod_perl.

Vorteil von mod_perl hier: die gesamte Stichwortliste mit Extradaten (eine Datenstruktur mit ca. 60 MB) kann im RAM gehalten werden.

http://www.assoziations-blaster.de/