forking PERL daemons

the "unixish" approach

the problem

- domain availability requests (many)
- no answers, no business (core service)
- typical .com code (sequential PHP)

first approach

- forking perl CGI
- better than PHP
- tons of processes on the web server

the first daemon

- prefork a population

```
for (1 .. $PREFORK){
    make_new_child();
}
```

- maintain the population

```
while (1) {
    sleep;
    for ($i = $children; $i < $PREFORK; $i++){
        make_new_child("net");
     }
}
```

forking

- make_new_child

```
sub make_new_child {
    die "fork: $!" unless defined ($pid = fork);
    if ($pid) {
        # parent stuff goes here (recording birth)
    } else {
        # do the client request
        exit;
    }
}
```

- you must not forget the exit!
- code like in perl cookbook

forking again

- signal handling gets messy
- zombies
- cron driven restarts

second approach

- Net::Server::PreFork
- writing just the functionality (hooks)
- working signal handling
- syslog, config file support, tcp wrapper, drop privileges, ...

drawbacks

- still one big daemon
- forking in a Net::Server daemon
- system outage while updates

current version

- many Net::Server daemons communicating with each other
- global error database
- local answer cache + timing database

config file

```
# user / group
user                    whois
group                   whois

# host / port
host                    127.0.0.1
port                    16000

# process control
min_servers             5
min_spare_servers       2
max_spare_servers       5
max_servers             20
max_requests            100
setsid                  1
pid_file                /Users/lenz/workspace/whoisd/var/run/whoisd.pid

# logging
log_file                /Users/lenz/workspace/whoisd/var/log/server.log
#log_file               Sys::Syslog
#syslog_logsock         unix
#syslog_ident           whoisd
#syslog_logopt          pid
log_level               4

# access control
#cidr_allow              62.146.33.0/24
#cidr_allow              82.135.96.210/29
#cidr_deny               192.168.0.12/30
daemon_name             whoisd
```

the daemon

```
use base qw(Net::Server::PreFork);

our ($VERSION) = '$Revision: 1.5 $' =~ m{\$Revision: \s+ (\S+)}xm;

### run the server
__PACKAGE__->run;
exit;
### set up some server parameters
sub configure_hook {
    my $self = shift;

    $self->{server_root} = "/tmp";
    $self->{server}->{conf_file} = 'etc/whoisd.das.conf';
    return;
}
# pre_loop_hook
sub pre_loop_hook {
    my $self = shift;
    $self->{start} = time();
    $self->{dbr}   = lib::DB::dbconnect("remote");
    $self->register();
    return;
}
sub process_request {
    my $self = shift;
    $self->{dbl} = lib::DB::dbconnect("local");
    my $stack;
    while (<STDIN>) {
        $stack .= $_;
    }
    print "accepted\n";
    my $serial = Data::Serializer->new( compress => 1 );
    $self->ask( $serial->deserialize($stack) );
    return;
}
```

features

- good performance
- easy maintainable
- less code (less errors)
- single daemons can be updated

drawbacks

- resource greedy
- strange lock ups under high load (rare but there)
- debugging not only your code