



The Future of mod_perl: Perl 6 and Beyond

Jeff Horwitz (jeff@smashing.org)

Frozen Perl

February 16, 2008



Who am I?

- System administrator and developer
 - Petfinder.com/Discovery Communications
- Parrot hacker
- Author of `mod_parrot` and `mod_perl6`



Where is mod_perl used?

- Used by ~4 million public domains*
- Internal web servers
 - company intranets
 - development

*Source: Netcraft



How is mod_perl used?

- CGI accelerator
- Custom handlers
- Applications
 - RT
 - MovableType
 - Slashcode
- Frameworks
 - Mason
 - Catalyst
 - AxKit
 - CGI::App



mod_perl myths we've heard

- mod_perl is just a faster CGI
- mod_perl isn't scalable
- mod_perl isn't used by any “real” sites



MYTHBUSTERS

- `mod_perl` is used and abused by many sites
 - Slashdot
 - Ticketmaster
 - Petfinder
 - 187 Million page views in January
 - Average usage: 70 page views per second
 - Peak usage: 130 page views per second
 - Even `www.frozen-perl.org` (ACT)



So what's the problem?

- Perl 6 is coming!
- No, really!
- Recent progress is scary
- Today's mod_perl will not work with Perl 6
 - different language
 - different backend VM (Parrot)
 - different embedding/extension API



So what, I can wait...

- Sure, but the Perl community can't
- Releasing Perl 6 without `mod_perl` alienates a HUGE audience
- This would set us back several years!
- Will lose marketshare and mindshare



Oh no! What can we do?

- Three possible strategies
 - Port mod_perl 2 to Perl 6
 - TOO HARD!
 - Write mod_perl6 using Parrot's embedding API
 - TOO EASY!
 - Write mod_parrot and layer Perl 6 on top
 - JUST RIGHT!



mod_parrot

- Embeds the Parrot VM in Apache
- Exposes the Apache API to Parrot
- Exposes Apache data structures to Parrot
- Gives Parrot High Level Languages (HLL) access to the Apache API
- Hides implementation details from the HLL



mod_perl6 on mod_parrot

- Parrot has a working Perl 6 compiler
 - Not finished, but maturing quickly
 - Adheres to the language spec (Synopses)
 - Passes sanity tests
 - OO support
- That means we can write mod_perl6 today!



mod_perl6 on mod_parrot

- mod_perl6 is an "HLL layer"
 - load Perl code
 - run Perl code
 - HLL specific configuration directives
 - supporting libraries (e.g. Apache::RequestRec)
 - return status
- Can be written in PIR (Parrot Intermediate Representation)

mod_perl6 in PIR

```
.namespace [ 'ModParrot'; 'HLL'; 'perl6' ]

.sub __onload :anon :load
    load_bytecode 'Languages/perl6/perl6.pbc'
    $P0 = new Hash
    set_hll_global 'loaded', $P0
.end

.sub load
    .param string handler
    .local string path, prefix, module_path
    .local string handler_sub
    .local pmc libdirs, env, handler_key, loaded
    loaded = get_hll_global 'loaded'
    $P0 = loaded[handler]
    handler_sub = 'handler'
    handler_key = split '::', handler
    unless_null $P0, RETURN_SUB
    module_path = join '/', handler_key
    env = new 'Env'
    $S0 = env['PERL6LIB']
    unless $S0, LOAD_HANDLER
    libdirs = split ':', $S0
    $I0 = elements libdirs
LIBDIR_LOOP:
    dec $I0
    prefix = libdirs[$I0]
    path = prefix
    path .= '/'
    path .= module_path
    path .= '.pm'
    $I1 = stat path, 0
        if $I1 goto LOAD_HANDLER
        if $I0 goto LIBDIR_LOOP
LOAD_HANDLER:
    $P0 = compreg 'Perl6'
    $P1 = $P0.'eval files' (path)
RETURN_SUB:
    $P1 = get_hll_global handler_key, handler_sub
    loaded[handler] = 1
    .return($P1)
.end

.sub handler
    .param pmc ctx
    .param string handler_name
    .local string handler_sub
    .local pmc handler
    .local pmc r
    .local pmc ap_const
    .local int status
    ap_const = get_root_global
        [ 'Apache'; 'Constants' ], 'ap_constants'
    r = ctx.'request_rec' ()
    handler_sub = 'handler'
    push_eh REPORT_ERROR
        handler = load(handler_name)
    pop_eh
    r.'content_type' ("text/html")

continued on next slide...
```



mod_perl6 in PIR

...continued from previous slide

```
    push_eh REPORT_ERROR
        status = handler(r)
    pop_eh
    goto RETURN_STATUS
REPORT_ERROR:
    get_results '0,0', $P0, $S0
    $S1 = handler_name
    concat $S1, ": "
    concat $S1, $S0
    $I0 = ap_const['APLOG_ERR']
    r.'log_rerror'(handler_name, 0, $I0, $S1)
    status = ap_const['HTTP_INTERNAL_SERVER_ERROR']
RETURN_STATUS:
    .return(status)
.end
```



YUCK!

PIR is Greek to most people.

Όλοι οι άνθρωποι γεννιούνται ελεύθεροι και ίσοι στην αξιοπρέπεια και τα δικαιώματα. Είναι προικισμένοι με λογική και συνείδηση, και οφείλουν να συμπεριφέρονται μεταξύ τους με πνεύμα αδελφοσύνης.

But we have something better...



mod_perl6 in Perl 6!

```
module ModParrot::HLL::perl6;

our %loaded_modules;

sub load($ctx, $module)
{
    unless (%loaded_modules{$module}) {
        use $handler;
        %loaded_modules{$module} = 1;
    }
}

sub handler($ctx, $name)
{
    my $r = $ctx.request_rec();
    load($ctx, $name);
    my $status = ::($name)::handler($r);
    return $status;
}
```

mod_perl6 in Perl 6!

```
module ModParrot::HLL::perl6;

our %loaded_modules;

sub load($ctx, $module)
{
    unless (%loaded_modules{$module}) {
        use $handler;
        %loaded_modules{$module} = 1;
    }
}

sub handler($ctx, $name)
{
    my $r = $ctx.request_rec();
    load($ctx, $name);
    my $status = ::($name)::handler($r);
    return $status;
}
```

mod_perl6 in Perl 6!

```
module ModParrot::HLL::perl6;

our %loaded_modules;

sub load($ctx, $module)
{
    unless (%loaded_modules{$module}) {
        use $handler;
        %loaded_modules{$module} = 1;
    }
}

sub handler($ctx, $name)
{
    my $r = $ctx.request_rec();
    load($ctx, $name);
    my $status = ::($name)::handler($r);
    return $status;
}
```

mod_perl6 in Perl 6!

```
module ModParrot::HLL::perl6;

our %loaded_modules;

sub load($ctx, $module)
{
    unless (%loaded_modules{$module}) {
        use $handler;
        %loaded_modules{$module} = 1;
    }
}

sub handler($ctx, $name)
{
    my $r = $ctx.request_rec();
    load($ctx, $name);
    my $status = ::($name)::handler($r);
    return $status;
}
```



mod_foo in Foo!

- Can write an HLL layer in the HLL if it:
 - supports namespaces
 - supports the Parrot object model
- Otherwise, a combination of PIR and HLL

Wow, mod_perl6 sounds great!

- But we don't need more vaporware
- So when can I try this?
- How about NOW?





Example 1: Polly

```
module ModPerl6::Polly;

sub handler($r) {
    $r.puts("SQUAWK! Polly says " ~ $r.args());
    0; # Apache OK
}
```




Example 1: Polly

```
# Apache configuration

ParrotAddHandler perl6 perl6-script


<Location /perl6/polly>
    SetHandler perl6-script
    ParrotHandler ModPerl6::Polly
</Location>
```



Example 2: Counter

```
module ModPerl6::Counter;

sub handler($r) {
    our $x;
    $x = 1 unless $x;
    $r.puts("Page views for this interpreter: $x\n");
    $x++;
    0; # Apache OK
}
```



Example 2: Counter

```
# Apache configuration

ParrotAddHandler perl6 perl6-script

<Location /perl6/counter>
    SetHandler perl6-script
    ParrotHandler ModPerl6::Counter
</Location>
```



Example 3: Authentication

```
module ModPerl6::MyAuthHandler;

sub handler($r) {
    my @arr = $r.get_basic_auth_pw();
    my $status = @arr[0];
    my $pw = @arr[1];
    my $rc;
    if ($pw eq 'squawk') {
        $rc = 0; # Apache OK
    }
    else {
        $rc = 401; # HTTP_UNAUTHORIZED
    }
    $rc;
}
```



Example 3: Authentication

```
# Apache configuration
```

```
<Directory /usr/local/apache2/htdocs/mptest>  
    ParrotLanguage perl6  
    ParrotAuthHandler ModPerl6::MyAuthHandler  
    Require valid-user  
</Directory>
```



What's Next?

- Add wrappers around mod_parrot libraries
 - Apache::RequestRec, etc.
- ModPerl6::Registry
 - Proof of concept works
- Support roadmapped mod_parrot features
 - HLL config directives (PerlHandler, etc.)
 - more data structures (server_rec, pools, etc.)
 - input & output filters



What about Perl 7? 8? ?

- If they're running on the Parrot VM...
 - Add new VM or Apache features to mod_parrot
 - Add another HLL layer for the new Perl
- If they're not running on Parrot...
 - You can find me in the nearest padded room



How can I help?

- Code
 - mod_parrot needs C programmers who know Apache
 - mod_perl6 needs mod_perl and Parrot programmers
 - Help write the Rakudo Perl 6 compiler
- Test
 - Write tests for mod_parrot and mod_perl6
 - Write tests for the Rakudo Perl 6 compiler
- Advocate
 - Blog, write articles, shout it from the rooftops
 - Debunk the FUD



Resources

- Me:
 - E-mail: jeff at smashing.org
 - IRC: jhorwitz in #parrot (irc.perl.org)
- mod_parrot: http://www.smashing.org/mod_parrot
- mod_perl6: http://www.smashing.org/mod_perl6
- Parrot: <http://www.parrotcode.org>
- Perl 6 Compiler (Rakudo): <http://rakudo.org>
- Perl 6 in general: <http://perlfoundation.org/perl6>



Copyright (c) 2008 Jeff Horwitz