Developing Web Applications with CGI::Application

Jason Purdy
jason@journalistic.com

Introduction

- Updated Slides will be made available
- Ask Questions Throughout

My WebDev Evolution

- Simple scripts
- CGI.pm scripts
- Separation of code & design
- MVC framework

CGI::Application

- Perl
- Minimal Web Application Framework
- FSA or Dispatch Table
- aka: CGIApp

CGIApp Benefits

- Great framework
- Over 5 Years Old
- Strong community
- Much much more...

What's it good for?

- Multi-step Web applications

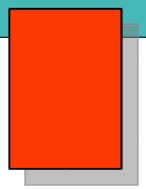
Development Approach

1. Mock-up Screens
2. Write Instance Script
3. Write CGI::App Frame PM
4. Code / Test / Debug
5. Roll Out

Typical Scenario: 3-Step Application

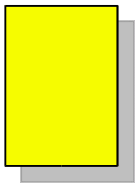
Mock-up Screens (write Template files)

user_data.TMPL



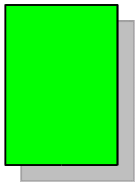
```
<html>
…
First Name:
<input type="text" name="fname" />
Last Name:
<input type="text" name="lname" />
<input type="hidden" name="rm" value="cc_data" />
…
</html>
```

cc_data.TMPL



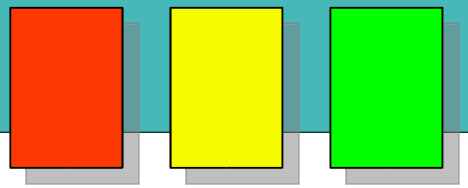
```
<html>
…
CC #:
<input type="text" name="cc_num" />
Exp. Date (MMYY):
<input type="text" name="cc_exp" />
<input type="hidden" name="fname"
  value="<!-- TMPL_VAR NAME="fname" ESCAPE=HTML -->" />
<input type="hidden" name="rm" value="receipt" />
…
</html>
```

receipt.TMPL



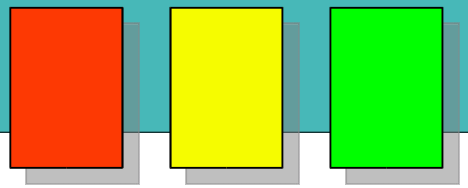
```
<html>
…
Thanks for your Order!
Ref Num:
<!-- TMPL_VAR NAME="ref_num" -->
…
</html>
```

Write Your Instance Script



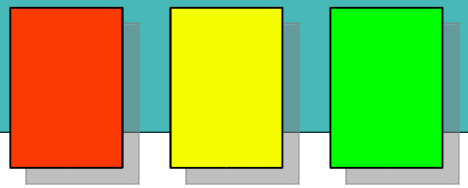
```
#!/usr/bin/perl -wT

# this is payme.cgi

$|++;
use strict;
use MyModule;

my $app = MyModule->new();
$app->run();
```

Write cgiapp Frame PM



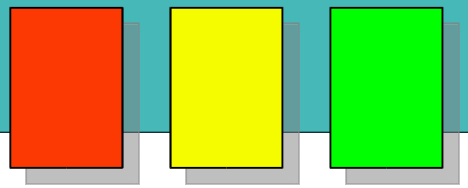
```
package MyModule;
use base 'CGI::Application';
sub setup {
    my ( $self );
    $self = shift;

    $self->run_modes( [
            'user_data', # red
            'cc_data',   # yellow
            'receipt'    # green
        ] );
    $self->mode_param( 'rm' );
    $self->start_mode( 'user_data' );
    $self->tmpl_path( '/path/to/your/template/files' );
    # $self->error_mode( 'error' );
    # Commented out, but good practice
}

1;
```

Flesh Out Individual Runmodes



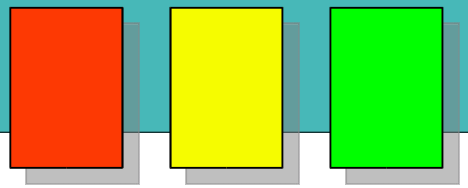
```
package MyModule;

sub setup { .. }

sub user_data {
    my ( $self, $template );

    $self = shift;
    $template = $self->load_tmpl( 'user_data.TMPL' );
    return $template->output();
}

1;
```

Flesh Out Individual Runmodes



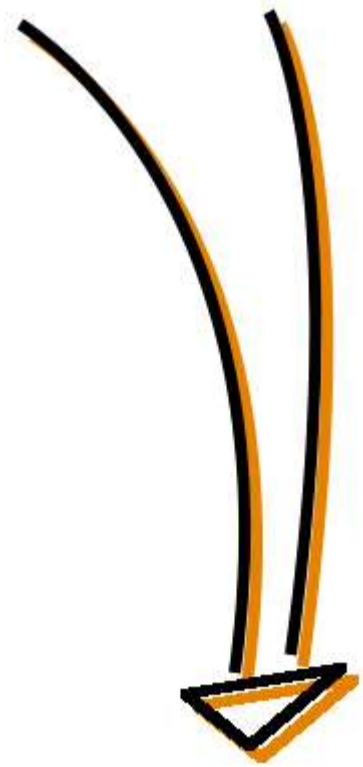
```
package MyModule;

sub setup { .. }

sub user_data { .. }

sub cc_data {
    my ( $self, $query, $template );

    $self = shift;
    $query = $self->query();
    $template = $self->load_tmpl(
              'cc_data.TMPL',
              'associate' => $query,
        );
    return $template->output();
}

1;
```

Flesh Out Individual Runmodes



```
package MyModule;

sub setup { .. }

sub user_data { .. }

sub cc_data { .. }

sub receipt {
    my ( $self, $query, $ref_num, $template );

    $self = shift;
    $query = $self->query();
    # Imaginary bill_card() method takes the CGI query object
    # and gets the data out of it and returns a $ref_num
    $ref_num = bill_card( $query );
    $template = $self->load_tmpl( 'receipt.TMPL' );
    $template->param( 'ref_num' => $ref_num );
    return $template->output();
}
1;
```

Understanding CGIApp's Method Flow



- cgiapp_init()
- setup()
- cgiapp_prerun()
- your runmodes
- cgiapp_postrun()
- teardown()

Gravy

- Easy to port to mod_perl
  - Anonymous handler in httpd.conf OR `handler()` method in your .PM class

What that looks like

Add `hander()` method to your subclass:

```
use Apache::Constants ':response';
sub handler ($$) {
     my ($pkg, $r) = @_;

     # Instantiate and run()
     # your CGI::Application module
     my $self = $pkg->new();
     $self->run();

     return OK;
}
```

Then tie into your configuration

- **Put your instance script in a Apache::Request directory:**

```
Alias /perl/ /usr/local/apache/perl/
<Location /perl>
        SetHandler      perl-script
        PerlHandler     Apache::Registry
        PerlSendHeader  On
        Options         +ExecCGI
</Location>
```

- **Or tie module directly in your Apache configuration:**

```
PerlModule My::CGIApp::Module
<Location /cgiapp>
        SetHandler      perl-script
        PerlHandler     My::CGIApp::Module
</Location>
```

Even Easier

- **Add an anonymous subroutine to your configuration:**

```
PerlModule My::CGIApp::Module
<Location /cgiapp>
        SetHandler      perl-script
        PerlHandler     "sub { My::CGIApp::Module->new()->run(); return OK; }"
</Location>
```

And Lastly...

- Check out CGI::Application::Plugin::Apache
- Creates Apache::Request object vs. a CGI object

Keep in mind

- Code changes will need apache restarts
- Template Caching Directives

Another Piece of Gravy

!!! Plugins !!!

I ♥ ValidateRM

- Very easy and powerful way to verify user data input (this leads into next talk [TU06])

How it affects runmodes

```
use CGI::Application::Plugin::ValidateRM;

sub user_data {
    my ( $self, $errs, $template );
    $self = shift;
    $errs = shift;
    # ...
    $template->param( $errs ) if $errs;
}

sub cc_data {
    my ( $self );

    $self = shift;
    my ( $results, $err_page ) = $self->check_rm(
        'user_data', '_dfv_profile' );
    return $err_page if $err_page;
}

1;
```

Then just create your dfv profile

```
use CGI::Application::Plugin::ValidateRM;

sub user_data { .. }

sub cc_data { .. }

sub _dfv_profile {
    return {
        'required' => [ qw( fname lname ) ]
    };
}

1;
```

Then for Sessions...

- **Simple integration:**

```
use CGI::Application::Plugin::Session;

sub runmode {
   my $self = shift;
   my $userid = $self->session->param( 'userid' );
   # ...
}
```

Easy to Customize Session Configuration

- **Provide your own customization in your cgiapp's `cgiapp_init` method:**

```
sub cgiapp_init {
   my $self = shift;
   # ...
   $self->session_config(
         CGI_SESSION_OPTIONS => [
            'driver:MySQL', undef, { Handle => $dbh }
                                  ],
         COOKIE_PARAMS        => {
            -expires  => '+3hrs',
            -domain    => 'example.com',
                                  },
      );
}
```

LogDispatch & `error_mode()`

- Use plugin
- Override cgiapp_init() to configure
- Bonus Points: Use error_mode!

```
use CGI::Application::Plugin::LogDispatch;

sub setup { ... }

sub cgiapp_init {
     my $self = shift;
     $self->log_config
     (
          APPEND_NEWLINE => 1,
          LOG_DISPATCH_MODULES =>
               [
                    {
                         module => 'Log::Dispatch::File',
                         name => 'debugging',
                         mode => 'append',
                         filename => '/path/to/log/file.log',
                         min_level => 'debug',
                         max_level => 'debug',
                    },
                    {
                         module => 'Log::Dispatch::File',
                         name => 'messages',
                         mode => 'append',
                         filename => '/path/to/log/another_file.log',
                         min_level => 'info',
                         max_level => 'alert',
                    },
                    {
                         module => 'Log::Dispatch::Email::MailSend',
                         name => 'email',
                         to => 'admin@example.com',
                         subject => '!!! Critical Error w/ My Web App !!!!',
                         min_level => 'emerg',
                    },
               ]
     );
}
```

Now in your runmodes, you can sprinkle in log messages:

```
$self->log->debug( “why isn't this working: VAR1 = $var” );
$self->log->info( sprintf( “%s deleted a record!”, $ENV{'REMOTE_USER'} ) );
```

```
# make sure you set this method name in your
# setup() using the error_mode() method:
#   $self->error_mode( 'error' );
sub error {
    my ( $self, $error, $admin_msg, $message );

    $self = shift;
    $error = $@;

    # Let's also log a critical error!
    $admin_msg = <<_EOF_;
Hey dude,

Just wanted to let you know that we got a critical error sent to the user:

Error Msg: $error
CGIApp Dump:
------------
_EOF_
    $admin_msg .= $self->dump();

    $self->log->emerg( $admin_msg );

    $message = <<_EOF_;
Hi There!

I'm sorry, but this application suffered a fatal error.  I have notified
developer in charge so they can take a look at it ASAP.  It is recommended
that you not try this again, as it may repeat the problem.  Feel free to
<a href=”lazy_programmer@example.com”>email the developer</a> to follow-up.
_EOF_

    $template = $self->load_tmpl( 'error.TMPL' );
    $template->output;
}
```

Other Gravy

- Lots of other plugin modules:
  - CAP::AnyTemplate
  - CAP::AutoRunMode
  - CAP::CompressGzip
  - CAP::Apache
  - CAP::Stream
  - CAP::TT
  - CAP::LogDispatch
  - CAP::HtmlTidy
  - CAP::DBH
  - CAP::BREAD

CAP = CGI::Application::Plugin

TIMTOWTDI

- Perl has >1 MVC Framework
  - Maypole/Catalyst
  - OpenInteract2
  - ~~CGI::Prototype~~
  - ~~CGI::MxScreen~~
- Your choice between flexibility/control and “Help”

Reference

CGIApp Wiki: http://cgi-app.org

CGI IRC: #cgiapp on irc.perl.org

Mailing Lists:

- **CGIApp:**

  cgiapp-subscribe@lists.erlbaum.net?Subject=Subscribe

- **HTMLTMPL:**

  http://lists.sourceforge.net/lists/listinfo/htmltmpl-support

- **DFV:**

  http://lists.sourceforge.net/lists/listinfo/cascade-dataform