

Bloggging with Catalyst

More Blogging Software?

Why?

Too hard to use!

Example

To compose a new post:

Open the web browser

Visit the Weblog

Click “New Post”

Type the post in a tiny
little text area.

Preview it.

Fix it.

Tweak it.

Finally. Done.

What's wrong with this?

Well, I already have a text editor.

emacs

vi

OpenOffice.org

I don't need ANOTHER!

What else is wrong?

Too hard to set up and
configure

RDBMS

MySQL

mod_perl

PHP

TLAs

No! I just want to write!

Finally, nobody else uses
threaded comments!

I want to have a
discussion,

not post mindless
“hey your post is cool”
replies.

Blame the software,

not the users.

The solution?

Firefox browser window titled "Hello, World - Firefox". The address bar shows "http://localhost:3000/articles/Hello" and the search bar contains "lorem ipsum". The browser toolbar includes navigation buttons and a search engine dropdown set to Google. The page content features a main heading "Test Weblog" and a sub-heading "Hello, World". A red diagonal banner in the top right corner reads "I ♥ YAPC::NA 2006". The post content includes a "Hello, YAPC::NA world" message, a "Lorem Ipsum" section, and a "Posted in: Tests" link. The footer shows "Done" and "Adblock" status.


File Edit View Go Bookmarks Tools Help

http://localhost:3000/articles/Hello Go lorem ipsum

Google Slashdot Weather Work Catalyst CPAN Search "Social" del.icio.us

Test Weblog

Not logged in. [Login](#).



Hello, World

Tags: [hello world](#) [+]

6-26-2006 (月) 9:13 pm

Hello, [YAPC::NA](#) world. This is a sample weblog posting. It has some text, some links, some code excerpts; everything a weblog needs. What it doesn't have, is a lot of text... so for that we'll paste in some Loerm Ipsum!

Home | [Articles](#)

Categories

- [CPAN::Modules](#)
- [Tests](#)

Tags

- [hello](#)
- [world](#)

Tags: [hello world](#) [+]

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent in pede. Quisque porta viverra neque. Vestibulum ut purus. Nunc imperdiet erat pellentesque pede. In hac habitasse platea dictumst. Integer et metus. Cras consectetur luctus orci. Nulla nisi nunc, varius nec, sollicitudin at, ornare ultricies, velit. In felis ante, porttitor a, porttitor nec, rutrum a, leo. Phasellus non odio ac nunc pellentesque sollicitudin. Aliquam erat volutpat. Nunc rhoncus mi id massa. Pellentesque aliquam eros id nulla. Etiam varius rutrum ante. Mauris interdum libero sit amet nisi. Aliquam erat volutpat.

Vestibulum condimentum porta urna. Ut malesuada. Aenean risus dolor, lobortis ac, tincidunt sed, tempus a, quam. Nulla nonummy nibh et turpis venenatis porta. Aenean nisl dolor, scelerisque nec, nonummy vitae, adipiscing in, justo. Integer diam. Nullam lectus. Cras ullamcorper posuere nulla. Nulla facilisi. Fusce nec diam. Fusce neque lacus, tincidunt quis, pulvinar ut, laoreet non, neque. Suspendisse nisl risus, pellentesque et, rhoncus a, congue eget, metus.

Posted in: [Tests](#)

Comments on [Hello, World](#) | 0 comments | [Post a comment](#)

Done S ✓ Adblock

How to post?

```
emacs@localhost.localdomain
File Edit Options Buffers Tools Perl Help
=pod

Hello, L<YAPC::NA> world. This is a sample weblog posting. It has
some text, some links, some C<code excerpts>; everything a weblog
needs. What it doesn't have is a lot of text... so for that we'll
paste in some Lorem Ipsum!

=head2 C<Lorem Ipsum>

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent in
pede. Quisque porta viverra neque. Vestibulum ut purus. Nunc imperdiet
erat pellentesque pede. In hac habitasse platea dictumst. Integer et
metus. Cras consectetur luctus orci. Nulla nisi nunc, varius nec,
sollicitudin at, ornare ultricies, velit. In felis ante, porttitor a,
porttitor nec, rutrum a, leo. Phasellus non odio ac nunc pellentesque
sollicitudin. Aliquam erat volutpat. Nunc rhoncus mi id
massa. Pellentesque aliquam eros id nulla. Etiam varius rutrum
ante. Mauris interdum libero sit amet nisi. Aliquam erat volutpat.

Vestibulum condimentum porta urna. Ut malesuada. Aenean risus dolor,
lobortis ac, tincidunt sed, tempus a, quam. Nulla nonummy nibh et
turpis venenatis porta. Aenean nisl dolor, scelerisque nec, nonummy
vitae, adipiscing in, justo. Integer diam. Nullam lectus. Cras
ullamcorper posuere nulla. Nulla facilisi. Fusce nec diam. Fusce neque
lacus, tincidunt quis, pulvinar ut, laoreet non, neque. Suspendisse
nisl risus, pellentesque et, rhoncus a, congue eget, metus.

--:-- Hello, World.pod All L10 (CPerl)----9:29PM 0.79-----
```

C-x C-s

Done.

Need to make some
changes?

C-x C-s

Done.

Don't like emacs?

Use OO.org, and save as
HTML.

[[Live Demonstration]]

What about commenting?

Even easier.

Click reply.

Type your message.

Sign it.

Done.

No registration.

No passwords.

No cookies.

Plus, your friends can
mathematically verify the
authenticity of your
message!

```
$ curl $LONG_URI | gpg --verify
```

```
gpg: Signature made Mon 26 Jun 2006  
09:38:39 PM CDT using RSA key ID  
DD25E42F
```

```
gpg: Good signature from "Jonathan T.  
Rockway <jrockway@uchicago.edu>"
```

Beat that, Slashdot!

How does this all work,
though?

Catalyst MVC

The View?

Template Toolkit.

```
[% WRAPPER page.tt %]  
  [% FOREACH article = articles %]  
    [% INCLUDE post.tt %]  
  [% END %]  
[% END %]
```

Yes, 5 lines of code to
generate HTML.

※ However, page.tt and post.tt are a bit longer. But not much.

And the Controller?

```
$c->stash->{template} =  
    q{blog_listing.tt};
```

```
$c->stash->{articles} =  
    [reverse sort $c->  
        Model('FileSystem')->  
            get_articles()];
```

Yes, 2 lines to feed the
data to the template.

※ However, there are some special cases that don't fit on the slide.

And the Model?

First, some philosophy.

A blog is basically some
text files that are
concatenated together.

Why involve
“record-breaking
performance and
scalability for transaction
processing and large scale
data warehouses” [1] !?

[1] <http://www.oracle.com/database/index.html>

Nobody reads your blog
anyway!

Seriously though.

Everything a RDBMS can
do, the filesystem can do
(better).

Let's see how.

First, what does the
database schema look like?

Posts:

Title, Content, [Metadata]

What does every file on
your filesystem look like?

Filename, Contents,

[Metadata]

(permissions, modification time, etc.)

See any similarities?

The filesystem is just a
database.

That comes with every
UNIX machine in
existence.

It's fast.

It's secure.

It's stable.

Why not use it for a data-driven application!?

How do we translate SQL
to this model?

Short answer: we don't.

We write our own brand-
new Catalyst model.

It has three methods.

```
get_articles  
get_categories  
get_tags
```

There are also two utility methods, `get_by_tag` and `get_by_category`; but these aren't strictly necessary.

`get_articles` finds all
files and calls
`Filesystem::Item->new`
for each.

(And returns a list of the `Items`.)

`get_tags` uses that list
of article lists to call
`$article->tags`
for each.

(And returns a list of the tags.)

`get_categories`
finds all subdirectories

(And returns a list of them.)

What about this
`Filesystem::Item`?

author

creation_time

modification_time

title

raw_text

text

signed

uri

Most of these methods
work the same way.

(There are more, too, but they do the same things – feed
some text to the template later on.)

```
sub title {
  my $self = shift;
  my $name;

  # use the title attribute if it exists
  eval {
    $name = getfattr($self->{path},
                    'user.title');
  };

  # otherwise the filename is more than
  # adequate
  if (!$name) {
    $name = $self->name;
    $name =~ s{[.] \w+$}{};
  }
  return $name;
}
```

etc. etc. until we have
everything we need.

Easy.

Once the model is up and running, it's easy to maintain, too.

Want to delete a post?

```
rm -f "The Post"
```

Add a category?

mkdir "New Category"

Move a post to that
category?

mv "Post" "Category"

DROP TABLE 'Posts'?

$r_m - r_f^*$

Why is this “easy to use”?

Because it works like every
other piece of software on
your system.

No need for additional
software.

No need to run new
services.

No need to learn SQL.

Back to Catalyst.
What about comments?

They're just subclasses of
`Filesystem::Item`.

That know how to attach
themselves to other
objects.

And that know how to
write themselves to the
disk.

Hey, isn't that a security
problem?

Letting arbitrary web
users write arbitrary files?

Of course.

How to avoid problems?

Don't use the user's input
for anything important.

The filename that stores the comment is generated entirely inside the program (a random number).

Since the web user can't control this, he can't do anything bad.

Everything else is stored
safely inside the file or in
an extended filesystem
attribute.

Where no characters have
special meaning.

(In filenames, characters like ‘/’ and ‘.’ have special meanings.)

So that's the MVC aspect.

Questions so far?

※ I have one. Are you ready to run out and replace Oracle with ext2fs?

We're also using some other cool technology.

Digital Signatures.

```
use Crypt::OpenPGP;
```

```
my $pgp = Crypt::OpenPGP  
        ->new;
```

```
$pgp->verify($message);
```

Crypt::OpenPGP deals
with all the details.

Dearmouing messages,
decompressing OpenPGP
packets, RSA, DSA.

No worries. Automatic.

It even retrieves keys from
the keyserver for us.

But actually, I have another Catalyst Model that caches keys and User information for us. See “Blog::User” and “Blog::Model::UserStore”.

Hot-pluggable text
formats.

The `Filesystem::Item`

knows what type of
document it is.

(text, html, pod, wiki, etc.)

When it needs to be converted to HTML for display, it simply calls on `Blog::Format` to handle the dirty work.

Blog::Format uses
Module::Pluggable to load
plugins.

It then asks each plugin,
“can you format \$type”?

The plugin returns a
number from 0 to 100.

100 means, “definitely”.

0 means, “nope, don’t
think so”.

After all the plugins have been interrogated, the numbers are sorted and the best formatter is selected.

If you write a new
formatter, you don't even
have to restart Catalyst
for the new format to
become available!

```
use Module::Pluggable (
    search_path => ['Blog::Format'],
    instantiate => 'new',
);

sub format {
    my ($text, $type) = @_;

    my @choices;
    foreach my $plugin (plugins()){
        if($plugin->can('can_format') && $plugin->can('format')){
            my $possibility = $plugin->can_format($type);
            push @choices, [$plugin, $possibility];
        }
    }

    # now sort the choices, and choose the highest
    @choices = sort {$b->[1] <=> $a->[1]} @choices;
    my $choice = $choices[0]->[0];

    return $choice->format($text, $type);
}
```

Easy!

Let's look at the HTML
formatter.

How does it work?

To prevent injection of
nasty HTML, we do a full
parse with
HTML::TreeBuilder

```
my $html =
    HTML::TreeBuilder->new;

$html->parse($text);
$html->eof;

my $result = $self->_parse
              ($html->guts);

$html->delete;

return $result;
```

```
sub _parse {
  foreach my $element (@elements) {
    ...
    if (blessed $element &&
        $element->isa('HTML::Element')) {
      my $tag = $element->tag;
      my @kids = $element->content_list;
      return "<$tag>" .
        _parse(@kids) .
        "</$tag>";
    }
    else {
      # element is just plain text.
      return _escape($element);
    }
  }
}
```

Parsing in this way lets us
make invalid HTML valid
XHTML.

And we can do cool things like converting `<h1>` tags to `<h3>` (and so on), so the outline of our document stays correct.

Example:

Save slashdot.org's main page in the blog storage directory.

Blog - Firefox

File Edit View Go Bookmarks Tools Help

http://localhost:3000/ Go

Google Slashdot Weather Work Catalyst Hello, World CPAN Search "Social" >>

Blog x Slashdot | PGP... x Slashdot | How... x Slashdot | Kent... x

- ◆ [Creative CommonsとFedora Project、共同でOpen Videoコンテンツを開催](#)
- ◆ [米Novell：企業向けSUSE Linuxを独習できるUSB外付けHDDを発売](#)

[Slashdot本家](#) ◆ [Supercomputer Models Sun's Corona Dynamics](#)

- ◆ [Supreme Court to Rule on 'Obvious' Patents](#)
- ◆ [OpenOffice.org Newspaper Ad Mockup Released](#)
- ◆ [DVD Format War Already Over?](#)
- ◆ [Canadian Gov't Gives Big Bucks to Copyright Lobby](#)
- ◆ [Linux Hackers Reclaim the WRT54G](#)
- ◆ [Kent State Banning Athletes from Using Facebook](#)
- ◆ [PGP & GPG](#)
- ◆ [Star Wars Galaxies Emulator Test Server Hits Alpha](#)
- ◆ [How to Win on Ebay: Snipe](#)

アレゲはアレゲ以上のなにもものでもなさげー アレゲ研究家

このページのすべての商標と著作権はそれぞれの所有者が有します。 コメントやユーザ日記に関しては投稿者が有します。
のこりのものは、 Copyright(C) 2001-2006 [OSDN](#) です。

[[ホーム](#) | [タレコむ](#) | [古いストーリー](#) | [FAQ](#) | [プライバシー](#) | [利用規約](#) | co-located at [SOFTBANK IDC](#) |]

[0 comments](#) | [Read more...](#)

Another Post!

Tags: [post](#) [blog](#) [+]

6-26-2006 (月) 10:40 pm

Hello, world. This is a **blog post!**

[0 comments](#) | [Read more...](#)

Hello, World

Tags: [hello world](#) [+]

6-26-2006 (月) 9:13 pm

Hello, [YAPC::NA](#) world. This is a sample weblog posting. It has some text, some links, some code excerpts; everything a weblog needs. What it doesn't have is a lot of text... so for that we'll paste in some Lorem Ipsum!

[http://rss.slashdot.org/Slashdot/slashdot?m=6253](#) S A Adblock

Did I mention it's fully
Unicode compliant?

(Thanks to `Catalyst::View::TT::ForceUTF8`.)

Are we out of time?

If not, let's look at a demo
again and tie up some
loose ends.

[[Demo Again]]

Topics: NonceStore, View::Dump, Sessions, and Catalyst::Plugin::Scheduler.

Final Points

Think Differently.

Don't trust the end-user.

終了

高橋さん ありがとうね。