NAME

HTML::Template::FAQ − Frequently Asked Questions about HTML::Template

SYNOPSIS

In the interest of greater understanding I've started a FAQ section of the perldocs. Please look in here before you send me email.

FREQUENTLY ASKED QUESTIONS

Is there a place to go to discuss HTML::Template and/or get help?

There's a mailing-list for discussing HTML::Template at html−template−users@lists.sourceforge.net. Join at:

```
http://lists.sourceforge.net/lists/listinfo/html-template-users
```

If you just want to get email when new releases are available you can join the announcements mailing-list here:

```
http://lists.sourceforge.net/lists/listinfo/html-template-announce
```

Is there a searchable archive for the mailing-list?

Yes, you can find an archive of the SourceForge list here:

```
http://dir.gmane.org/gmane.comp.lang.perl.modules.html-template
```

I want support for <TMPL_XXX>! How about it?

Maybe. I definitely encourage people to discuss their ideas for HTML::Template on the mailing list. Please be ready to explain to me how the new tag fits in with HTML::Template's mission to provide a fast, lightweight system for using HTML templates.

NOTE: Offering to program said addition and provide it in the form of a patch to the most recent version of HTML::Template will definitely have a softening effect on potential opponents!

I found a bug, can you fix it?

That depends. Did you send me the VERSION of HTML::Template, a test script and a test template? If so, then almost certainly.

If you're feeling really adventurous, HTML::Template is publicly available on GitHub (https://github.com/mpeters/html−template). Please feel free to fork it and send me a pull request with any changes you have.

<TMPL_VAR>s from the main template aren't working inside a <TMPL_LOOP>! Why?

This is the intended behavior. `<TMPL_LOOP>` introduces a separate scope for `<TMPL_VAR>`s much like a subroutine call in Perl introduces a separate scope for `my` variables.

If you want your `<TMPL_VAR>`s to be global you can set the `global_vars` option when you call `new()`. See above for documentation of the `global_vars` `new()` option.

How can I pre-load my templates using cache-mode and mod_perl?

Add something like this to your startup.pl:

```
use HTML::Template;
use File::Find;

print STDERR "Pre-loading HTML Templates...\n";
find(
    sub {
        return unless /\.tmpl$/;
        HTML::Template->new(
            filename => "$File::Find::dir/$_",
            cache    => 1,
        );
    },
    '/path/to/templates',
    '/another/path/to/templates/'
);
```

Note that you'll need to modify the `return unless` line to specify the extension you use for your template files – I use *.tmpl*, as you can see. You'll also need to specify the path to your template files.

One potential problem: the */path/to/templates/* must be **EXACTLY** the same path you use when you call `HTML::Template->new()`. Otherwise the cache won't know they're the same file and will load a new copy – instead getting a speed increase, you'll double your memory usage. To find out if this is happening set `cache_debug` = 1> in your application code and look for "CACHE MISS" messages in the logs.

What characters are allowed in TMPL_* names?

Numbers, letters, '.', '/', '+', '–' and '_'.

How can I execute a program from inside my template?

Short answer: you can't. Longer answer: you shouldn't since this violates the fundamental concept behind HTML::Template – that design and code should be separate.

But, inevitably some people still want to do it. If that describes you then you should take a look at HTML::Template::Expr. Using HTML::Template::Expr it should be easy to write a `run_program()` function. Then you can do awful stuff like:

```
    <tmpl_var expr="run_program('foo.pl')">
```

Just, please, don't tell me about it. I'm feeling guilty enough just for writing HTML::Template::Expr in the first place.

What's the best way to create a <select> form element using HTML::Template?

There is much disagreement on this issue. My personal preference is to use CGI.pm's excellent `popup_menu()` and `scrolling_list()` functions to fill in a single `<tmpl_var select_foo>` variable.

To some people this smacks of mixing HTML and code in a way that they hoped HTML::Template would help them avoid. To them I'd say that HTML is a violation of the principle of separating design from programming. There's no clear separation between the programmatic elements of the `<form>` tags and the layout of the `<form>` tags. You'll have to draw the line somewhere – clearly the designer can't be entirely in charge of form creation.

It's a balancing act and you have to weigh the pros and cons on each side. It is certainly possible to produce a `<select>` element entirely inside the template. What you end up with is a rat's nest of loops and conditionals. Alternately you can give up a certain amount of flexibility in return for vastly simplifying your templates. I generally choose the latter.

Another option is to investigate HTML::FillInForm which some have reported success using to solve this problem.