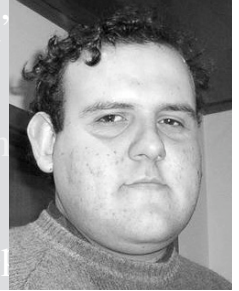


Templating My Output

Separating code from presentation

by Alberto Simões
ams@di.uminho.pt



One of the various acronyms for Perl is Practical Extraction and Report Language. Thus, Perl should have good resources for creating reports. While Perl has a built-in reporting facility called Perl formats (see the *perform* man page) I will not talk about that in this article.

I will talk about `HTML::Template`, `Text::Template`, and `Template Toolkit`, three templating modules available on CPAN. While I can template almost anything using `Text::Template` I will show how to use `HTML::Template` because its syntax is more HTML friendly and `Template Toolkit` because it is very powerful.

The idea behind the use of templates is the generation of text (or XML, HTML, or whatever) where some portions are dynamic and change according to some data stored somewhere. For a simple idea, consider a mailing tool that send a personalized email to a list of persons, changing the name of the person in each mail by using a template to specify how these emails will look, and where to place each person's data in the text.

All of these modules work in a similar manner when writing templates: I just write the text I want, being it plain text or HTML, and put special placeholder marks where the data should appear.

■ `HTML::Template` -----

`HTML::Template`, by Sam Tregar, is a templating module for HTML files. I just write the HTML file and place some special tags where the dynamic values should be placed. As a simple (and typical) example, imagine I want to say Hello to different folks I know, creating a web page for each. The page should look something like:

```
<html>
<head>
  <title>Saying hello to brian</title>
</head>

<body>
  <h1>Hello, brian!</h1>
</body>
</html>
```

Now, if I want to create a template for these files, I would replace “brian” with a placeholder that I can fill in later with whichever name I choose:

```
<html>
<head>
```

```
  <title>Saying hello to <TMPL_VAR
NAME=FOLK></title>
</head>

<body>
  <h1>Hello, <TMPL_VAR NAME=FOLK>!</h1>
</body>
</html>
```

And to create a file for each of my friends I would need to write the following code:

```
#!/usr/bin/perl
use HTML::Template;

my $template = HTML::Template->new(
  filename => 'template.html',
);

@friends = qw/larry brian damian/;

for my $friend (@friends) {
  $template->param(FOLK => $friend);

  open H, ">$friend.html"
    or warn "$friend: $!";
  print H $template->output;
  close H;
}
```

For each friend, the template value is different and I send the output to an HTML file. Now I have a personalized page for each friend.

The interesting thing about `HTML::Template` is that it supports loops and conditionals. This way it is simple to generate lists of items or to use variables conditionally. For instance, if I have a list of food I want each of my friends to bring for the party, I can change the template to include a list:

```
<html>
<head>
  <title>Saying hello to <TMPL_VAR
NAME=FOLK></title>

</head>

<body>
```