HTML::Template で DB 擬似連携

2006 年 3 月 19 日(日)
はしもと <ksk@be.to>

前回に続いて、HTML::Template の filter の使い方について、サンプルをいくつか紹介します。

filter とは？

HTML::Template オブジェクトを生成するときに filter オプションとして、サブルーチンのリファレンスを渡すことができます。

filter 用サブルーチンには、HTML::Template が読み込んだテンプレートファイルの内容がリファレンスとして渡されます。
HTML::Template がテンプレートファイルを解析するより先に filter 処理が実行されます。
filter で処理した結果が最終的に HTML::Template の書式に合えばいいので、渡されたリファレンスを書き換えて様々なカスタマイズを行うことができます。
渡されるリファレンスは、スカラーリファレンスと配列リファレンス(行単位の配列)を指定することができますが、ここではスカラーリファレンスしか取り扱いません。

詳しくは perldoc を参照してください。
http://perldoc.jp/docs/modules/HTML-Template-2.6/HTML/Template.pod

DB 擬似連携

詳細はサンプルコードを見てください。

要はカスタムタグを作って、カスタムタグがあればデータベースからデータを読み込んでテンプレートオブジェクトに設定する、というだけの単純なものです。

３番目のサンプルは、HTML::Template::Expr を使用しています。
４番目のサンプルとして、テンプレートに記述した SQL を実行させるものを用意してみました。

sample1.tmpl
```
<ul>
<TMPL_LANG_LOOP>
      <li><TMPL_VAR NAME='name' ESCAPE='HTML'></li>
</TMPL_LANG_LOOP>
</ul>
```

sample2.tmpl
```
<TMPL_LANG_LIST>
```

sample3.tmpl

```
<ul>
<TMPL_LANG_LOOP>
      <li<TMPL_IF EXPR="id == user_lang"> class="current"</TMPL_IF>>
            <TMPL_VAR NAME='name' ESCAPE='HTML'>
      </li>
</TMPL_LANG_LOOP>
</ul>
```

sample4.tmpl

```
<h1>lang list</h1>
<ul>
<TMPL_QUERY_LOOP QUERY='SELECT * FROM "lang" ORDER BY name, id'>
      <li><TMPL_VAR NAME='name' ESCAPE='HTML'></li>
</TMPL_QUERY_LOOP>
</ul>

<h1>user list</h1>
<ul>
<TMPL_QUERY_LOOP QUERY='SELECT * FROM "users" ORDER BY id'>
      <li><TMPL_VAR NAME='name' ESCAPE='HTML'></li>
</TMPL_QUERY_LOOP>
</ul>
```

sample.pl

```
#!/usr/bin/env perl

use strict;
use warnings;

use CGI;
use DBI;
use Data::Dumper;
use HTML::Template;
use HTML::Template::Expr;
use Jcode;

use constant  DATA_SOURCE =>  'dbi:Pg:dbname=test';
use constant  DB_USER     =>  'www';
use constant  DB_PASSWORD =>  undef;
use constant  DB_SCHEMA   =>  'kansaipm';

MAIN:{
  my $mode    = 4;
  my $user_id = 2;
  my $db      = connect_db();
  my $user    = get_user( $db, $user_id );
  my $tmpl;

  $tmpl = load_tmpl1( 'sample1.tmpl', $db ) if $mode == 1;
  $tmpl = load_tmpl2( 'sample2.tmpl', $db ) if $mode == 2;
  $tmpl = load_tmpl3( 'sample3.tmpl', $db ) if $mode == 3;
  $tmpl = load_tmpl4( 'sample4.tmpl', $db ) if $mode == 4;

  # ユーザーデータHashにPrefixを付けてテンプレートに登録
  $tmpl->param( map { ( "user_$_" => $user->{$_} ) } keys %$user );
```

```
  # 表示の都合でShift_JISに変換
  print Jcode::convert( $tmpl->output, 'sjis', 'euc' );

  $db->disconnect;
}

sub load_tmpl1  {
  my $file  = shift;
  my $db    = shift;
  my $use_lang_list;
  my $lang_list_key = '__lang_list';

  my $lang_list_filter  = sub {
    my $text_ref  = shift;

    # カスタムタグを通常のテンプレートタグに置換
    if( $$text_ref =~ s/<TMPL_LANG_LOOP>/<TMPL_LOOP NAME='$lang_list_key'>/
ig ) {
      $$text_ref =~ s!</TMPL_LANG_LOOP>!</TMPL_LOOP>!ig;

      # 後でデータを登録するためにフラグを設定
      $use_lang_list = 1;
    }
  };

  my $tmpl  = HTML::Template->new(
    filename    =>  $file,
    filter      =>  $lang_list_filter,  # ←フィルター設定
    global_vars =>  1,
    die_on_bad_params =>  0,
  );

  if( $use_lang_list )  {
    # データ登録
    $tmpl->param( $lang_list_key => lang_list( $db ) );
  }

  return $tmpl;
}

sub load_tmpl2  {
  my $file  = shift;
  my $db    = shift;
  my $use_lang_list;
  my $lang_list_key = '__lang_list';

  my $lang_list = <<TMPL;
<ul>
<TMPL_LOOP NAME="$lang_list_key">
  <li><TMPL_VAR NAME='name' ESCAPE='HTML'></li>
</TMPL_LOOP>
</ul>
TMPL

  my $lang_list_filter  = sub {
    my $text_ref  = shift;

    # カスタムタグをテンプレートに置き換え
    if( $$text_ref =~ s/<TMPL_LANG_LIST>/$lang_list/ig )  {
```

```
      # 後でデータを登録するためにフラグを設定
      $use_lang_list  = 1;
    }
  };

  my $tmpl  = HTML::Template->new(
    filename    =>  $file,
    filter      =>  $lang_list_filter,  # ←フィルター設定
    global_vars =>  1,
    die_on_bad_params =>  0,
  );

  if( $use_lang_list )  {
    # データ登録
    $tmpl->param( $lang_list_key => lang_list( $db ) );
  }

  return $tmpl;
}

# load_tmpl1との違いは
# HTML::TemplateではなくHTML::Template::Exprを使っているところです
sub load_tmpl3  {
  my $file  = shift;
  my $db    = shift;
  my $use_lang_list;
  my $lang_list_key = '__lang_list';

  my $lang_list_filter  = sub {
    my $text_ref  = shift;

    if( $$text_ref =~ s/<TMPL_LANG_LOOP>/<TMPL_LOOP NAME='$lang_list_key'>/
ig ) {
      $$text_ref =~ s!</TMPL_LANG_LOOP>!</TMPL_LOOP>!g;
      $use_lang_list  = 1;
    }
  };

  # HTML::Template::Exprを使用
  my $tmpl  = HTML::Template::Expr->new(
    filename    =>  $file,
    filter      =>  $lang_list_filter,
    global_vars =>  1,
    die_on_bad_params =>  0,
  );

  if( $use_lang_list )  {
    $tmpl->param( $lang_list_key => lang_list( $db ) );
  }

  return $tmpl;
}

sub load_tmpl4  {
  my $file  = shift;
  my $db    = shift;
  my $count = 0;
  my %params;
```

```
  my $query_loop_filter = sub {
    my $text_ref  = shift;

    # ここの正規表現はかなり適当です(^^;
    $$text_ref  =~  s{
      <TMPL_QUERY_LOOP\s+QUERY=([\'\"])(.+?)\1\s*>
    }{
      my $key   = sprintf "__query_loop%d", ++$count;
      my $query = CGI::unescapeHTML( $2 );
      $params{$key} = get_list_by_query( $db, $query );
      "<TMPL_LOOP NAME='$key'>";
    }xieg;
    $$text_ref  =~  s!</TMPL_QUERY_LOOP>!</TMPL_LOOP>!g;
  };

  my $tmpl  = HTML::Template->new(
    filename    =>  $file,
    filter      =>  $query_loop_filter,
    global_vars =>  1,
    die_on_bad_params =>  0,
  );

  $tmpl->param( %params );

  return $tmpl;
}

sub connect_db  {
  my $db    = DBI->connect( DATA_SOURCE, DB_USER, DB_PASSWORD );
  if( DB_SCHEMA ) {
    $db->do( sprintf( 'SET SEARCH_PATH TO "%s"', DB_SCHEMA ) );
  }
  return $db;
}

sub lang_list {
  my $db    = shift;
  my $query = qq! SELECT * FROM "lang" ORDER BY id!;
  return $db->selectall_arrayref( $query, { Columns => {} } );
}

sub get_user  {
  my $db    = shift;
  my $id    = shift;
  my $query = qq! SELECT * FROM "users" WHERE id = ?!;
  return $db->selectrow_hashref( $query, undef, $id );
}

sub get_list_by_query {
  my $db    = shift;
  my $query = shift;
  return $db->selectall_arrayref( $query, { Columns => {} } );
}
```

最後に

今回紹介した方法は、多分 MovableType のカスタムタグでも同じように使っていると思います。

４番目のサンプルは SQL をテンプレートに記述するというちょっと強引なやり方ですが、カスタムタグの定義の仕方を工夫すればもう少し簡単な指定でデータを取り出せると思います。

でも…。

HTML::Template でここまでやるぐらいなら、素直に Template::Toolkit を使った方がいいかも(^^;。

おまけ

http://www.scrapcode.net/perl/

こんなサイトを作ろうとしています。