Template Toolkit

A Fast, Flexible Template System for Perl

Introduction

- Introduction
  - What is Template Toolkit
  - Why TT is Useful
- Using Template Toolkit
- Resources

What is Template Toolkit?

- Fast, Extensible Templating System
- Written in Perl
- Stable, well established, since 1996.
  Last update 7 February 2012.

Why Templates?

- Templates separate your calculation from your display logic.
- Easier to modify templates than code which generates formatted output.

Example 1: Ugly Code

```
print "Between $start and $end, there were $redCount attempts by ".
"<A HREF=\"http://www.symantec.com/avcenter/venc/data/codered.ii.html\">Code Red</A>, "
. "and $nimdaCount attempts by the "
. "<A HREF=\"http://www.symantec.com/avcenter/venc/data/w32.nimda.a\@mm.html\">Nimda worm</a>"
. " to infect this system.\n";

print "<p>Code Red had " . scalar (keys %redIPs) . " unique IP addresses:</p>\n";

my $ip = "";

print "<TABLE ROWS=\"" . scalar (keys %redIPs) . "\" COLS=\"2\" BORDER=\"1\">\n";
print "\t<TR><TH>IP Address:<TH>Attacks:</TR>\n";
foreach $ip (keys %redIPs)
{
      print "\t<TR><TD>$ip<TD>". $redIPs{$ip} ."\n";
}
print "</TABLE>\n";
```

Example 2a: Clean

```
use FindBin::Bin;
use Template;
my $tt = Template->new({
      INCLUDE_PATH = “$FindBin::Bin/templates”,
      INTERPOLATE = 1,
});

my $vars = {};
# Build the same variables you had before,
# but put them in a hashref.
$vars->{...} = ...

$tt->process(‘virusreport.tt’, $vars);
```

Example 2b: Template

```
Between [% start %] and [% end %], there were [% redCount %] attempts by
<A HREF="http://www.symantec.com/avcenter/venc/data/codered.ii.html">Code Red</A>,
and [% nimdaCount %] attempts by the
<A HREF="http://www.symantec.com/avcenter/venc/data/w32.nimda.a\@mm.html">Nimda worm</a>
to infect this system.

<p>Code Red had [% redIPs.keys.size %] unique IP addresses:</p>

<TABLE ROWS="[%- redIps.keys.size -%]" COLS="2" BORDER="1">
<TR><TH>IP Address:<TH>Attacks:</TR>
[% FOREACH ip = redIPs.keys.sort %]
      <TR><TD>[% ip %]</TD><TD>[% redIPs.$ip %]</TD></TR>
[% END %]
</TABLE>
```

Data Builds Output

- Same data can build multiple templates:
  - HTML receipt page
  - E-mail receipt with HTML part and a plain text part
  - CSV, TSV, XML, JSON, etc.

Using Template Toolkit

The Basics

- Build Data
  - Everything in a hashref
  - Create TT object
    - Specify location for templates
  - Process Template

The Data

- Hashref
  - Scalars for single items
  - Array refs to loop over
  - Hashrefs to reference
  - Objects to call

Scalars

- A scalar item in the data hash is a single replacement variable.

```
my $dataHash = {
        thing => ‘foo!’,
        person => ‘Fred’,
        selector = 12,
}
```

```
This is a [% thing %]
You are [% person %]
You selected [% selector %].
```

```
This is a foo!
You are Fred
You selected 12 .
```

Array References

- An array reference can be looped over, processing each item.

```
my $dataHash = {
        person => ‘Fred’,
        cars => [
            ‘Ferrari’, ‘Tesla’,
           ‘Lotus’, ‘Porsche’
        ],
}
```

```
[% person %] dreams of owning:
[% FOREACH dreamcar = cars %]
    A brand-new [% dreamcar %]
[% END %]
```

```
Fred dreams of owning:
   A brand-new  Ferrari
   A brand-new  Tesla
   A brand-new  Lotus
   A brand-new Porsche
```

Hash References

- Hash references can hold complex data items. Can also iterate over keys.

```
my $dataHash = {
        person => {
            name => 'Fred',
            hat_size => 38,
            ear_depth => 11,
            wears_cologne => 0,
        },
}
```

```
[% person.name %] has a hat size of [%
person.hat_size %].

We know the following things about this
person:
[% FOREACH key = person.keys %]
   [% key %]: [% person.$key %]
[% END %]
```

```
Fred  has a hat size of  38 .

We know the following things about this
person:
   hat_size: 38
   wears_cologne: 0
   ear_depth: 11
   name: Fred
```

Objects

- Objects in the data hash can have methods or data items accessed.

```
my $dataHash = {
   uri = URI->new("http://www.example.com:1235/home/index.cgi",
}
```

```
The URI [% uri.as_string %] points to a [% uri.scheme %] on [% uri.host %].
```

The URI http://www.example.com:1235/home/index.cgi points to a http on www.example.com.

Create TT Object

- Creates like any other object.
- Lots of helpful initializers.
- Very configurable.

```
use FindBin::Bin;
use Template;
my $tt = Template->new({
        INCLUDE_PATH = “$FindBin::Bin/templates”,
        INTERPOLATE = 1,
});
```

Use TT Object

- Call methods to set up, process.
- Several ways to process.

```
# Prints to stdout
$tt->process('virusreport.tt', $vars);

open(my $outfh, '|-', 'mail -s "virus report"')
   or die ("Can't launch mail!: $!");
$tt->process('virusemail.tt', $vars, $outfh);
close($outfh);

my $text = '';
$tt->process('virusemail.tt', $vars, $text);

sub reporter {
      my $text = shift;
}

$tt->process('virusemail.tt', $vars, \&reporter);
```

TT Details

- TT Formatting
- Accessing items
- Commands

TT Formatting

- [% %] defines a TT region
- TT commands are in UPPERCASE. Variables are in lower or mixed case.
- Commands can span lines. Start w/ command, end with [% END %]

White Space and TT

- White space is significant, and left alone.
- “[% something %]” will replace with “_text _” because there are spaces on either side of the variable name.
- [%- or -%] can eat the whitespace.
  - An option can make that the default.

Accessing Items

- Hashrefs in the data item can be accessed with a “.” instead of an arrow.
- Methods can be called with a “.”.
- “Virtual Methods” can be called with a “.”.
- It’s all dots to TT.

Control Flow

- IF/ELSE/END
- FOREACH/END
- WHILE/END
- SWITCH/CASE

IF/ELSE/END

- Select which block to emit.

```
my $dataHash = {
        person => ‘Fred’,
        tall = 0,
}
```

```
You are [% person %]
[% IF count %]
   You are tall!
[% ELSE %]
    You’re not tall.
[% END %]
```

```
You are Fred
   You’re not tall.
```

FOREACH/END

- Process each item in an array.

```
my $dataHash = {
        person => ‘Fred’,
        cars => [
            ‘Ferrari’, ‘Tesla’,
           ‘Lotus’, ‘Porsche’
        ],
}
```

```
[% person %] dreams of owning:
[% FOREACH dreamcar = cars %]
    A brand-new [% dreamcar %]
[% END %]
```

```
Fred dreams of owning:
   A brand-new  Ferrari
   A brand-new  Tesla
   A brand-new  Lotus
   A brand-new Porsche
```

WHILE/END

- Loop as long as an expression is true.

```
my $dataHash = {
};
bless $dataHash, __PACKAGE__;
sub maybetrue {
    return 1 if(rand > 0.5);
    return 0;
}
```

```
[% WHILE maybetrue %]
It was true!
[% END %]
```

```
It was true!
```

A random number of times!

SWITCH/CASE

```
my $dataHash = {
        color = ‘blue’,
}
```

```
[% SWITCH color %]
[%CASE ‘red’ %]
Red Wine
[% CASE ‘blue’ %]
Blue Cheese
[% CASE ‘green’ %]
Green Beans
[% END %]
```

```
Blue Cheese
```

Truthiness

- Works just like Perl!
  - undef, '' or 0 is false
  - anything else is true

Building Components

- Include separate .tt files
- Define reusable blocks of template

BLOCK

- Define a template component to use with INCLUDE, PROCESS, and WRAPPER
- No output until used later.

```
[% BLOCK row %]
<tr><th>[% header %]</th><td>[% data %]</td</tr>
[% END %]
```

INCLUDE

- Process and emit the results of a BLOCK or another file
  - Searches for a file if no matching BLOCK name
  - Localizes changes made in the block

```
my $dataHash = {
    header => 'Name',
    data => 'Fred',
}
```

```
[% BLOCK row %]
<tr><th>[% header %]</th><td>[% data
%]</td></tr>
[% END %]

<table>
[% INCLUDE row %]
</table>
```

```
<table>
<tr><th> Name </th><td> Fred </td></tr>
</table>
```

PROCESS

- Process and emit the results of a BLOCK or another file
  - Does NOT changes made in the block

```
my $dataHash = {
    header => 'Name',
    data => 'Fred',
}
```

```
[% BLOCK row %]
<tr><th>[% header %]</th><td>[% data
%]</td></tr>
[% END %]

<table>
[% INCLUDE row %]
</table>
```

```
<table>
<tr><th> Name </th><td> Fred </td></tr>
</table>
```

WRAPPER

- Process and emit the results of a BLOCK or another file
  - Takes parameters!

```
[% BLOCK row %]
<tr><th>[% header %]</th><td>[% data %]</td></tr>
[% END %]

<table>
[% PROCESS row header = ‘Sky’ data = ‘blue’ %]
[% PROCESS row header = ‘Trees’ data = ‘green’ %]
</table>
```

```
<table>
<tr><th> Sky </th><td> blue </td></tr>
<tr><th> Trees </th><td> green </td></tr>
</table>
```

VMethods

- Virtual Methods you can call on certain data types to do common operations.
- Called by a dot and the method name after the variable.
- There's a bunch of them.
- Can be chained together.

```
<table class="vert_table">
[% INCLUDE row label='Title' value=form.title name='title' size=80 %]
[% INCLUDE row label='Artist' value=form.artist name='artist' size=80 %]
[% INCLUDE row label='Cover Condition' value=form.cover_condition name='cover_condition' list=conditions %]
[% INCLUDE row label='Media Condition' value=form.media_condition name='media_condition' list=conditions %]
[% INCLUDE row label='Description' value=form.description name='description' size=16000 cols=80 rows=10 %]
[% INCLUDE row label='Label' value=form.label name='label' size=10 %]
[% INCLUDE row label='Release Date' value=form.release_date name='release_date' size=5 %]
[% INCLUDE row label='Release Number' value=form.release_number name='release_number' size=10 %]
[% INCLUDE row label='Release Country' value=form.release_country name='release_country' size=10 %]
[% INCLUDE row label='Mono or Stereo' value=form.mono_stereo name='mono_stereo' list=mono_stereo_types %]
[% INCLUDE row label='Number of Records' value=form.nrecinset name='nrecinset' size=5 %]
[% INCLUDE row label='On Hand' value=form.onhand name='onhand' size=10 %]
[% IF c.user_exists %]
[% INCLUDE row label='Location' value=form.physical_location name='physical_location' size=35 %]
[% END %]
</table>

[% BLOCK row %]
<tr>
 <th>[% label %]:</th>
 [% IF edit %]
  [% IF rows and cols %]
   <td><textarea name="[%- name -%]" maxlength="[%- size -%]" rows="[%- rows -%]" cols="[%- cols -%]">[%- value -%]</textarea></
td>
  [% ELSIF list %]
    <td>
      <select name="[%- name -%]">
        [% FOREACH item IN list %]
        <option [%- IF item == value -%] selected [%- END -%]>[% item %]</option>
        [% END %]
      </select>
    </td>
  [% ELSE %]
   <td><input name="[%- name -%]" type="text" size="[%- size -%]" value="[%- value -%]"></input></td>
  [% END %]
 [% ELSE %]
 <td>[% value %]</td>
 [% END %]
</tr>
[% END %]
```

Resources

- http://www.template-toolkit.org
  - Excellent site, excellent manuals.
- Perldoc
- O'reilly book, "Perl Template Toolkit"