Template::Flute - Modern HTML and PDF Engine

Stefan Hornburg (Racke)
`racke@linuxia.de`

Perl Mongers Hamburg, 4th April 2011

Why

- Separation of web design and programming
- How available template engines violate this principle
  - Mini language (Template::Toolkit)
  - Inline code
  - CSS selectors (HTML::Zoom)
- Solutions by Template::Flute
  - Static HTML file
  - Specification file
- Further Goals
  - Great flexibility
  - Tweaks through tree manipulations

Cart: Hash

```
$cart = [
        {isbn => '978-0-2016-1622-4',
         title => 'The Pragmatic Programmer',
         quantity => 1, price => 49.95},

        {isbn => '978-1-4302-1833-3',
         title => 'Pro Git',
         quantity => 1, price => 34.99},
];
```

Cart: HTML Template

```
<table class="cart">
<tr class="cartheader">
<th>Name</th><th>Quantity</th><th>Price</th>
</tr>
<tr class="cartitem">
<td class="name">Perl 6</td>
<td><input class="quantity" name="quantity" size="3" value="10"></td>
<td class="price">$$$</td>
</tr>
<tr class="cartheader">
<th colspan="2"></th><th>Total</th></tr>
<tr>
<td colspan="2"></td><td class="cost">$$$</td></tr>
</table>
```

Cart: ITL

```
<table class="cart">
<tr class="cartheader">
<th>Name</th><th>Quantity</th><th>Price</th>
</tr>
[item-list]
<tr class="cartitem">
<td class="name">[item-modifier title]</td>
<td><input class="quantity" name="quantity" size="3" value="[item-qua
<td class="price">[item-price]</td>
</tr>
[/item-list]
<tr class="cartheader">
<th colspan="2"></th><th>Total</th></tr>
<tr>
<td colspan="2"></td><td class="cost">[total-cost]</td>
</tr>
</table>
```

Cart: Template::Toolkit

```
<table class="cart">
<tr class="cartheader">
<th>Name</th>
<th>Quantity</th>
<th>Price</th>
</tr>
[% FOREACH cart %]
<tr class="cartitem">
<td class="name">[% title %]</td>
<td><input class="quantity" name="quantity" size="3" value="[% quanti
<td class="price">[% price %]</td>
</tr>
[% END %]
</table>
```

Cart: HTML::Zoom

```
use HTML::Zoom;

$cart = ...
$zoom = HTML::Zoom->from_file('cart.html');

$zoom = $zoom->select('.cartitem')->repeat_content([
  map { my $field = $_; sub {
    $_[0]->select('.name')->replace_content($field->{title});
    $_[0]->select('.quantity')->replace_content($field->{quantity});
    $_[0]->select('.price')->replace_content($field->{price});
    };
  } @$cart]);

$zoom = $zoom->select('.cost')->replace_content(49.95 + 34.99);

print $zoom->to_html();
```

Template Problems

- Mini language in HTML template
- Dynamic pages (border cases, errors, ...)

Template::Flute Concept

- Specification
- Template
- Data or objects (iterator)

Template::Flute Specification (XML)

```
<specification name="cart" description="Cart">
<list name="cart" class="cartitem" iterator="cart">
<param name="name" field="title"/>
<param name="quantity"/>
<param name="price"/>
</list>
<value name="cost"/>
</specification>
```

Template::Flute Specification (Config::Scoped)

```
list cart {
   class = cartitem
   iterator = cart
}
param quantity {
   list = cart
}
param price {
   list = cart
}
param name {
   list = cart
   field = title
}
value cost {
   name = cost
```

Template::Flute Script (XML)

```
use Template::Flute;

my ($cart, $flute, %values);

$cart = ...
$values{cost} = ...

$flute = new Template::Flute(specification_file => 'cart.xml',
                             template_file => 'cart.html',
                             iterators => {cart => $cart},
                             values => \%values,
);

print $flute->process();
```

Template::Flute Script (Config::Scoped)

```
use Template::Flute;

my ($cart, $flute, %values);

$cart = ...
$values{cost} = ...

$flute = new Template::Flute(specification_file => 'cart.conf',
                             specification_parser => 'Scoped',
                             template_file => 'cart.html',
                             iterators => {cart => $cart},
                             values => \%values,
);

print $flute->process();
```

Menus: Database table

```
CREATE TABLE menus (
  code int NOT NULL auto_increment,
  name varchar(255) NOT NULL DEFAULT '',
  url  varchar(255) NOT NULL DEFAULT '',
  menu_name varchar(64) NOT NULL DEFAULT '',
  permission varchar(64) NOT NULL DEFAULT '',
  weight int NOT NULL DEFAULT 0,
  PRIMARY KEY(code),
  KEY(menu_name)
);
```

Menus: Specification

```
<specification name="menu" description="Menu">
<list name="menu" class="menu" table="menus">
<input name="name" required="1" field="menu_name"/>
<param name="label" field="name"/>
<param name="url"/>
</list>
</specification>
```

Menus: Template

```
<ul class="menu">
<li><a href="" class="url"><span class="label"></span></li>
</ul>
```

Menus: Script

```
use Template::Flute;
use Template::Flute::Database::Rose;

$db_object = new Template::Flute::Database::Rose(dbname => 'temzoo',
    dbtype => 'Pg',
);

$flute = new Template::Flute(specification_file => 'menu.xml',
   template_file => 'menu.html',
   database => $db_object,
  );

$flute->process({name => main});
```

Iterators

- next method
- count method
- hash reference as return value

Template::Flute::Iterator

```
use Template::Flute::Iterator;

Template::Flute::Iterator->new($cart);
```

Subclassing Template::Flute::Iterator

```
package MyApp::Iterator;

use base 'Template::Flute::Iterator';

sub new {
    ...
    $self->seed([...]);
    return $self;
}
```

Elements

- value
- list
- param
- input
- container
- form
- i18n
- sort

Lists with alternating rows

```
<table class="cart">
<tr class="cartheader">
<th>Name</th><th>Quantity</th><th>Price</th>
</tr>
<tr class="cartitem odd">
<td class="name">Perl 6</td>
<td><input class="quantity" name="quantity" size="3" value="10"></td>
<td class="price">$$$</td>
</tr>
<tr class="cartitem even">
<td class="name">Pro Git</td>
<td><input class="quantity" name="quantity" size="3" value="10"></td>
<td class="price">$$$</td>
</tr>
</table>
```

Filter: Specification

```
<specification name="menu" description="Menu">
<list name="menu" class="menu" table="menus">
<input name="name" required="1" field="menu_name"/>
<param name="label" field="name"/>
<param name="url" target="href" filter="link"/>
</list>
</specification>
```

Filter: Function

```
sub link_filter {
    my $page = shift;
    my $url;

     $url = ...

     return $url;
}

$flute = new Template::Flute(specification_file => 'menu.xml',
   template_file => 'menu.html',
   database => $db_object,
   filters => {link => \&link_filter},
  );
```

Global Filter

```
<specification name="menu" description="Menu">
<filter name="acl" field="permission"/>
<list name="menu" class="menu" table="menus">
<input name="name" required="1" field="menu_name"/>
<param name="label" field="name"/>
<param name="url" target="href" filter="link"/>
</list>
</specification>
```

Sort: Specification

```
<specification name="menu" description="Menu">
<list name="menu" class="menu" table="menus">
<sort name="default">
<field name="weight" direction="desc"/>
<field name="code" direction="asc"/>
</sort>
<input name="name" required="1" field="menu_name"/>
<param name="label" field="name"/>
<param name="url" target="href" filter="link"/>
</list>
</specification>
```

I18N

```
sub translate {
    my $text = shift;

    ...

    return $text;
}

$i18n = new Template::Flute::I18N (\&translate);

$flute = new Template::Flute(specification_file => 'menu.xml',
   template_file => 'menu.html',
   database => $db_object,
   i18n => $i18n,
  );
```

I18N: Lookup Keys

```
<i18n name="returnurl" key="RETURN_URL"/>
```

Forms: Specification

```
<specification name='search' description=''>
<form name='search'>
<field name='searchterm'/>
<field name='searchsubmit'/>
</form>
</specification>
```

Forms: Manipulating

set_action	Changing form action
set_method	Changing form method (GET, POST)
fill	Fill form fields

HTML to PDF

- HTML template processing
- PDF conversion (PDF::API2)
  1. calculate
  2. partition
  3. render
- Inline CSS

PDF: Code

```
$flute = new Template::Flute (specification_file => 'pdf.xml',
                              template_file => 'pdf.html',
                              values => \%values);
$flute->process();

$pdf = new Template::Flute::PDF (template => $flute->template(),
                                 file => 'invoice.pdf');
$pdf->process();
```

PDF: Import

```
$import{file} = 'shippinglabel.pdf';
$import{scale} = 0.8;
$import{margin} = {left => '3mm', top => '6mm'};

$pdf = new Template::Flute::PDF (template => $flute->template(),
                                 file => 'invoice.pdf',
                                 import => \%import);
```

Dancer Example

```
use Dancer;

get '/' => sub {
    return 'Hello world';
};

dance;
```

Fruits Demo

```
dancer -a Fruits
$EDITOR Fruits/lib/Fruits.pm
```

(see next slide)

```
Fruits/bin/app.pl
```

Fruits Demo

```
package Fruits;

prefix '/fruits';

# route for image files
get '/*.jpg' => sub {
    my ($name) = splat;

    send_file "images/$name.jpg";
};

# route for fruits page
get qr{/?(?<page>.*)}  => sub {
    template 'fruits';
};
```

Dancer & Template::Flute

```
template: "template_flute"

engines:
  template_flute:
    iterators:
      fruits:
        class: JSON
        file: fruits.json
```

Current and Future Use Cases

- Very Large Product Lists
- Shop Backend
  - Product Editor
  - Product Search & Replace
- PDF Invoices
- Template Engine for Interchange

Roadmap

- Documentation
- Tests
- Conditions
- Empty lists, number of results
- Selected items
- Paging
- Trees

The End

Git	git://git.linuxia.de/temzoo.git
CPAN	`http://search.cpan.org/dist/Template-Flute/`
	`http://search.cpan.org/dist/Template-Flute-PDF/`
	`http://search.cpan.org/dist/Dancer-Template-TemplateFlute/`
Talk	`http://www.linuxia.de/talks/hhmongers2011/tf-hhmongers2011-beamer.pdf`
Questions	???