Tilt Templates

(See https://github.com/rtomayko/tilt/blob/master/docs/TEMPLATES.md for a rendered, HTML-version of this file).

While all Tilt templates use the same basic interface for template loading and evaluation, each varies in its capabilities and available options. Detailed documentation on each supported template engine is provided below.

There are also some file extensions that have several implementations (currently ERB and Markdown). These template classes have certain features which are guaranteed to work across all the implementations. If you wish to be compatible with all of these template classes, you should only depend on the cross-implementation features.

- ERB - Generic ERB implementation (backed by erb.rb or Erubis)
- erb.rb - Tilt::ERBTemplate
- Erubis - Tilt::ErubisTemplate
- Haml - Tilt::HamlTemplate
- Liquid - Tilt::LiquidTemplate
- Nokogiri - Tilt::NokogiriTemplate
- Builder - Tilt::BuilderTemplate
- Markaby - Tilt::MarkabyTemplate
- Radius - Tilt::RadiusTemplate

Tilt also includes support for CSS processors like LessCSS and Sass, CoffeeScript and some simple text formats.

- Less - Tilt::LessTemplate
- Sass - Tilt::SassTemplate
- Scss - Tilt::ScssTemplate
- CoffeeScript - Tilt::CoffeeScriptTemplate
- Textile - Tilt::RedClothTemplate
- Creole - Tilt::CreoleTemplate
- RDoc - Tilt::RDocTemplate

Tilt has extensive support for Markdown, backed by one of four different implementations (depending on which are available on your system):

- Markdown - Generic Markdown implementation
- RDiscount - Tilt::RDiscountTemplate
- Redcarpet - Tilt::RedcarpetTemplate
- BlueCloth - Tilt::BlueClothTemplate
- Kramdown - Tilt::KramdownTemplate
- Maruku - Tilt::MarukuTemplate

ERB (erb, rhtml)

ERB is a simple but powerful template languge for Ruby. In Tilt it's backed by Erubis (if installed on your system)

or by erb.rb (which is included in Ruby's standard library). This documentation applies to both implementations.

Example

```
Hello <%= world %>!
```

Usage

ERB templates support custom evaluation scopes and locals:

```
>> require 'erb'
>> template = Tilt.new('hello.html.erb')
>> template.render(self, :world => 'World!')
=> "Hello World!"
```

Or, use Tilt['erb'] directly to process strings:

```
template = Tilt['erb'].new { "Hello <%= world %>!" }
template.render(self, :world => 'World!')
```

Options

:trim => trim

Omits newlines and spaces around certain lines (usually those that starts with <% and ends with %>). There isn't a specification for how trimming in ERB should work, so if you need more control over the whitespace, you should use erb.rb or Erubis directly.

:outvar => '_erbout'

The name of the variable used to accumulate template output. This can be any valid Ruby expression but must be assignable. By default a local variable named _erbout is used.

erb.rb (erb, rhtml)

ERB implementation available in Ruby's standard library.

All the documentation of ERB applies in addition to the following:

Usage

The Tilt::ERBTemplate class is registered for all files ending in .erb or .rhtml by default, but with a *lower* priority than ErubisTemplate. If you specifically want to use ERB, it's recommended to use #prefer:

```
Tilt.prefer Tilt::ERBTemplate
```

NOTE: It's suggested that your program require 'erb' at load time when using this template engine within a

threaded environment.

Options

:trim => true

The ERB trim mode flags. This is a string consisting of any combination of the following characters:

- '>' omits newlines for lines ending in >
- '<>' omits newlines for lines starting with <% and ending in %>
- '%' enables processing of lines beginning with %
- true is an alias of <>

:safe => nil

The $SAFE level; when set, ERB code will be run in a separate thread with $SAFE set to the provided level.

:outvar => '_erbout'

The name of the variable used to accumulate template output. This can be any valid Ruby expression but must be assignable. By default a local variable named _erbout is used.

See also

- ERB documentation

Erubis (erb, rhtml, erubis)

Erubis is a fast, secure, and very extensible implementation of ERB.

All the documentation of ERB applies in addition to the following:

Usage

The Tilt::ErubisTemplate class is registered for all files ending in .erb or .rhtml by default, but with a *higher* priority than ERBTemplate. If you specifically want to use Erubis, it's recommended to use #prefer:

```
Tilt.prefer Tilt::ErubisTemplate
```

NOTE: It's suggested that your program require 'erubis' at load time when using this template engine within a threaded environment.

Options

:engine_class => Erubis::Eruby

Allows you to specify a custom engine class to use instead of the default which is Erubis::Eruby.

:escape_html => false

When true, Erubis::EscapedEruby will be used as the engine class instead of the default. All content within <%= %> blocks will be automatically html escaped.

:outvar => '_erbout'

The name of the variable used to accumulate template output. This can be any valid Ruby expression but must be assignable. By default a local variable named _erbout is used.

:pattern => '<% %>'

Set pattern for embedded Ruby code.

:trim => true

Delete spaces around <% %>. (But, spaces around <%= %> are preserved.)

See also

- Erubis Home
- Erubis User's Guide

Haml (haml)

Haml is a markup language that's used to cleanly and simply describe the HTML of any web document without the use of inline code. Haml functions as a replacement for inline page templating systems such as PHP, ASP, and ERB, the templating language used in most Ruby on Rails applications. However, Haml avoids the need for explicitly coding HTML into the template, because it itself is a description of the HTML, with some code to generate dynamic content. (more)

Example

```
%html
  %head
    %title= @title
  %body
    %h1
      Hello
      = world + '!'
```

Usage

The Tilt::HamlTemplate class is registered for all files ending in .haml by default. Haml templates support custom evaluation scopes and locals:

```
>> require 'haml'
>> template = Tilt.new('hello.haml')
=> #<Tilt::HamlTemplate @file='hello.haml'>
```

```
>> @title = "Hello Haml!"
>> template.render(self, :world => 'Haml!')
=> "
<html>
  <head>
    <title>Hello Haml!</title>
  </head>
  <body>
    <h1>Hello Haml!</h1>
  </body>
</html>"
```

Or, use the Tilt::HamlTemplate class directly to process strings:

```
>> require 'haml'
>> template = Tilt::HamlTemplate.new { "%h1= 'Hello Haml!'" }
=> #<Tilt::HamlTemplate @file=nil ...>
>> template.render
=> "<h1>Hello Haml!</h1>"
```

NOTE: It's suggested that your program require 'haml' at load time when using this template engine within a threaded environment.

Options

Please see the Haml Reference for all available options.

See also

- #haml.docs
- Haml Tutorial
- Haml Reference

Liquid (liquid)

Liquid is for rendering safe templates which cannot affect the security of the server they are rendered on.

Example

```
<html>
  <head>
    <title>{{ title }}</title>
  </head>
  <body>
    <h1>Hello {{ world }}!</h1>
  </body>
```

```
</html>
```

Usage

Tilt::LiquidTemplate is registered for all files ending in .liquid by default. Liquid templates support locals and objects that respond to #to_h as scopes:

```
>> require 'liquid'
>> require 'tilt'
>> template = Tilt.new('hello.liquid')
=> #<Tilt::LiquidTemplate @file='hello.liquid'>
>> scope = { :title => "Hello Liquid Templates" }
>> template.render(nil, :world => "Liquid")
=> "
<html>
  <head>
    <title>Hello Liquid Templates</title>
  </head>
  <body>
    <h1>Hello Liquid!</h1>
  </body>
</html>"
```

Or, use Tilt::LiquidTemplate directly to process strings:

```
>> require 'liquid'
>> template = Tilt::LiquidTemplate.new { "<h1>Hello Liquid!</h1>" }
=> #<Tilt::LiquidTemplate @file=nil ...>
>> template.render
=> "<h1>Hello Liquid!</h1>"
```

NOTE: It's suggested that your program require 'liquid' at load time when using this template engine within a threaded environment.

See also

- Liquid for Programmers
- Liquid Docs
- GitHub: Shopify/liquid

Radius (radius)

Radius is the template language used by Radiant CMS. It is a tag language designed to be valid XML/HTML.

Example

```
<html>
<body>
  <h1><r:title /></h1>
  <ul class="<r:type />">
  <r:repeat times="3">
    <li><r:hello />!</li>
  </r:repeat>
  </ul>
  <r:yield />
</body>
</html>
```

Usage

To render a template such as the one above.

```
scope = OpenStruct.new
scope.title = "Radius Example"
scope.hello = "Hello, World!"

require 'radius'
template = Tilt::RadiusTemplate.new('example.radius', :tag_prefix=>'r')
template.render(scope, :type=>'hlist'){ "Jackpot!" }

# The result will be:

    <html>
    <body>
      <h1>Radius Example</h1>
      <ul class="hlist">
        <li>Hello, World!</li>
        <li>Hello, World!</li>
        <li>Hello, World!</li>
      </ul>
      Jackpot!
    </body>
    </html>
```

See also

- Radius
- Radiant CMS

Textile (textile)

Textile is a lightweight markup language originally developed by Dean Allen and billed as a "humane Web text generator". Textile converts its marked-up text input to valid, well-formed XHTML and also inserts character entity references for apostrophes, opening and closing single and double quotation marks, ellipses and em dashes.

Textile formatted texts are converted to HTML with the RedCloth engine, which is a Ruby extension written in C.

Example

```
h1. Hello Textile Templates

Hello World. This is a paragraph.
```

Usage

NOTE: It's suggested that your program require 'redcloth' at load time when using this template engine in a threaded environment.

See Also

- RedCloth

RDoc (rdoc)

RDoc is the simple text markup system that comes with Ruby's standard library.

Example

```
= Hello RDoc Templates

Hello World. This is a paragraph.
```

Usage

NOTE: It's suggested that your program require 'rdoc', require 'rdoc/markup', and require 'rdoc/markup/to_html' at load time when using this template engine in a threaded environment.

See also

- RDoc

Markdown (markdown, md, mkd)

Markdown is a lightweight markup language, created by John Gruber and Aaron Swartz. For any markup that is not covered by Markdown's syntax, HTML is used. Marking up plain text with Markdown markup is easy and Markdown formatted texts are readable.

Markdown formatted texts are converted to HTML with one of these libraries:

- RDiscount - Tilt::RDiscountTemplate
- Redcarpet - Tilt::RedcarpetTemplate
- BlueCloth - Tilt::BlueClothTemplate
- Kramdown - Tilt::KramdownTemplate
- Maruku - Tilt::MarukuTemplate

Tilt will use fallback mode (as documented in the README) for determining which library to use. RDiscount has highest priority - Maruku has lowest.

Example

```
Hello Markdown Templates
========================

Hello World. This is a paragraph.
```

Usage

To wrap a Markdown formatted document with a layout:

```
layout = Tilt['erb'].new do
  "<!doctype html><title></title><%= yield %>"
end
data = Tilt['md'].new { "# hello tilt" }
layout.render { data.render }
# => "<!doctype html><title></title><h1>hello tilt</h1>\n"
```

Options

Every implementation of Markdown *should* support these options, but there are some known problems with the Kramdown and Maruku engines.

:smartypants => true|false

Set true to enable Smarty Pants style punctuation replacement.

In Kramdown this option only applies to smart quotes. It will apply a subset of Smarty Pants (e.g. ... to ...) regardless of any option.

Maruku ignores this option and always applies smart quotes (and nothing else).

:escape_html => true|false

Set true disallow raw HTML in Markdown contents. HTML is converted to literal text by escaping < characters.

Kramdown and Maruku doesn't support this option.

See also

- Markdown Syntax Documentation

RDiscount (markdown, md, mkd)

Discount is an implementation of the Markdown markup language in C. RDiscount is a Ruby wrapper around Discount.

All the documentation of Markdown applies in addition to the following:

Usage

The Tilt::RDiscountTemplate class is registered for all files ending in .markdown, .md or .mkd by default with the highest priority. If you specifically want to use RDiscount, it's recommended to use #prefer:

```
Tilt.prefer Tilt::RDiscountTemplate
```

NOTE: It's suggested that your program require 'erubis' at load time when using this template engine within a threaded environment.

See also

- Discount
- RDiscount
- GitHub: rtomayko/rdiscount