tornado.template — Flexible output generation

A simple template system that compiles templates to Python code.

Basic usage looks like:

```
t = template.Template("<html>{{ myvalue }}</html>")
print(t.generate(myvalue="XXX"))
```

`Loader` is a class that loads templates from a root directory and caches the compiled templates:

```
loader = template.Loader("/home/btaylor")
print(loader.load("test.html").generate(myvalue="XXX"))
```

We compile all templates to raw Python. Error-reporting is currently... uh, interesting. Syntax for the templates:

```
### base.html
<html>
  <head>
    <title>{% block title %}Default title{% end %}</title>
  </head>
  <body>
    <ul>
      {% for student in students %}
        {% block student %}
          <li>{{ escape(student.name) }}</li>
        {% end %}
      {% end %}
    </ul>
  </body>
</html>

### bold.html
{% extends "base.html" %}

{% block title %}A bolder title{% end %}

{% block student %}
  <li><span style="bold">{{ escape(student.name) }}</span></li>
{% end %}
```

Unlike most other template systems, we do not put any restrictions on the expressions you can include in your statements. `if` and `for` blocks get translated exactly into Python, so you can do complex expressions like:

```
{% for student in [p for p in people if p.student and p.age > 23] %}
  <li>{{ escape(student.name) }}</li>
{% end %}
```

Translating directly to Python means you can apply functions to expressions easily, like the `escape()` function in the examples above. You can pass functions in to your template just like any other variable (In a `RequestHandler`, override `RequestHandler.get_template_namespace`):

```
### Python code
def add(x, y):
   return x + y
template.execute(add=add)

### The template
{{ add(1, 2) }}
```

We provide the functions `escape()`, `url_escape()`, `json_encode()`, and `squeeze()` to all templates by default.

Typical applications do not create `Template` or `Loader` instances by hand, but instead use the `render` and `render_string` methods of `tornado.web.RequestHandler`, which load templates automatically based on the `template_path` `Application` setting.

Variable names beginning with `_tt_` are reserved by the template system and should not be used by application code.

Syntax Reference

Template expressions are surrounded by double curly braces: `{{ ... }}`. The contents may be any python expression, which will be escaped according to the current autoescape setting and inserted into the output. Other template directives use `{% %}`.

To comment out a section so that it is omitted from the output, surround it with `{# ... #}`.

These tags may be escaped as `{{!`, `{%!`, and `{#!` if you need to include a literal `{{`, `{%`, or `{#` in the output.

`{% apply *function* %}...{% end %}`

Applies a function to the output of all template code between `apply` and `end`:

```
{% apply linkify %}{{name}} said: {{message}}{% end %}
```

Note that as an implementation detail apply blocks are implemented as nested functions and thus may interact strangely with variables set via `{% set %}`, or the use of `{% break %}` or `{% continue %}` within loops.

`{% autoescape *function* %}`

Sets the autoescape mode for the current file. This does not affect other files, even those referenced by `{% include %}`. Note that autoescaping can also be configured globally, at the `Application` or `Loader`.:

```
{% autoescape xhtml_escape %}
{% autoescape None %}
```

`{% block *name* %}...{% end %}`

Indicates a named, replaceable block for use with `{% extends %}`. Blocks in the parent template will be replaced with the contents of the same-named block in a child template.:

```
<!-- base.html -->
<title>{% block title %}Default title{% end %}</title>

<!-- mypage.html -->
{% extends "base.html" %}
{% block title %}My page title{% end %}
```

`{% comment ... %}`

A comment which will be removed from the template output. Note that there is no `{% end %}` tag; the comment goes from the word `comment` to the closing `%}` tag.

`{% extends *filename* %}`

Inherit from another template. Templates that use `extends` should contain one or more `block` tags to replace content from the parent template. Anything in the child template not contained in a `block` tag will be ignored. For an example, see the `{% block %}` tag.

`{% for *var* in *expr* %}...{% end %}`

Same as the python `for` statement. `{% break %}` and `{% continue %}` may be used inside the loop.

`{% from *x* import *y* %}`

Same as the python `import` statement.

`{% if *condition* %}...{% elif *condition* %}...{% else %}...{% end %}`

Conditional statement - outputs the first section whose condition is true. (The `elif` and `else` sections are optional)

`{% import *module* %}`

Same as the python `import` statement.

`{% include *filename* %}`

Includes another template file. The included file can see all the local variables as if it were copied directly to the point of the `include` directive (the `{% autoescape %}` directive is an exception). Alternately, `{% module Template(filename, **kwargs) %}` may be used to include another template with an isolated namespace.

`{% module *expr* %}`

Renders a `UIModule`. The output of the `UIModule` is not escaped:

```
{% module Template("foo.html", arg=42) %}
```

`UIModules` are a feature of the `tornado.web.RequestHandler` class (and specifically its `render` method) and will not work when the template system is used on its own in other contexts.

`{% raw *expr* %}`

Outputs the result of the given expression without autoescaping.

`{% set *x* = *y* %}`
: Sets a local variable.

`{% try %}...{% except %}...{% else %}...{% finally %}...{% end %}`
: Same as the python `try` statement.

`{% while *condition* %}... {% end %}`
: Same as the python `while` statement. `{% break %}` and `{% continue %}` may be used inside the loop.

`{% whitespace *mode* %}`
: Sets the whitespace mode for the remainder of the current file (or until the next `{% whitespace %}` directive). See `filter_whitespace` for available options. New in Tornado 4.3.

Class reference

class **`tornado.template.Template`**(*template_string, name="<string>", loader=None, compress_whitespace=None, autoescape="xhtml_escape", whitespace=None*) [source]

A compiled template.

We compile into Python from the given template_string. You can generate the template from variables with generate().

Construct a Template.

Parameters:
- **template_string** (*str*) – the contents of the template file.
- **name** (*str*) – the filename from which the template was loaded (used for error message).
- **loader** (*tornado.template.BaseLoader*) – the `BaseLoader` responsible for this template, used to resolve `{% include %}` and `{% extend %}` directives.
- **compress_whitespace** (*bool*) – Deprecated since Tornado 4.3. Equivalent to `whitespace="single"` if true and `whitespace="all"` if false.
- **autoescape** (*str*) – The name of a function in the template namespace, or `None` to disable escaping by default.
- **whitespace** (*str*) – A string specifying treatment of whitespace; see `filter_whitespace` for options.

Changed in version 4.3: Added `whitespace` parameter; deprecated `compress_whitespace`.

`generate`(***kwargs*) [source]

Generate this template with the given arguments.

class **`tornado.template.BaseLoader`**(*autoescape='xhtml_escape', namespace=None, whitespace=None*) [source]

Base class for template loaders.

You must use a template loader to use template constructs like `{% extends %}` and `{% include %}`. The loader caches all templates after they are loaded the first time.

Construct a template loader.

Parameters:
- **autoescape** (*str*) – The name of a function in the template namespace, such as "xhtml_escape", or `None` to disable autoescaping by default.
- **namespace** (*dict*) – A dictionary to be added to the default template namespace, or `None`.
- **whitespace** (*str*) – A string specifying default behavior for whitespace in templates; see `filter_whitespace` for options. Default is "single" for files ending in ".html" and ".js" and "all" for other files.

Changed in version 4.3: Added `whitespace` parameter.

reset() [source]

Resets the cache of compiled templates.

resolve_path(*name*, *parent_path=None*) [source]

Converts a possibly-relative path to absolute (used internally).

load(*name*, *parent_path=None*) [source]

Loads a template.

class **tornado.template.Loader**(*root_directory*, ***kwargs*) [source]

A template loader that loads from a single root directory.

class **tornado.template.DictLoader**(*dict*, ***kwargs*) [source]

A template loader that loads from a dictionary.

exception **tornado.template.ParseError**(*message*, *filename=None*, *lineno=0*) [source]

Raised for template syntax errors.

`ParseError` instances have `filename` and `lineno` attributes indicating the position of the error.

Changed in version 4.3: Added `filename` and `lineno` attributes.

tornado.template.filter_whitespace(*mode*, *text*) [source]

Transform whitespace in `text` according to `mode`.

Available modes are:

- `all`: Return all whitespace unmodified.
- `single`: Collapse consecutive whitespace with a single whitespace character, preserving newlines.
- `oneline`: Collapse all runs of whitespace into a single space character, removing all newlines in the process.

New in version 4.3.