Flask-Genshi Documentation

Release 0.1

Dag Odenhall

September 14, 2011

CONTENTS

Flask-Genshi is an extension to Flask that allows you to easily use Genshi for templating.

Features:

- Render to a `response_class` based on template filename extension and set mimetype and doctype accordingly
- Easy switching of rendering methods, i.e., output HTML as HTML4, XHTML, HTML5...
- Genshi filters with a syntax fit for Flask
- Flask Jinja globals, filters and tests
- `flask.Module` templates loading
- Flask context processors
- Rendering from strings

Not quite yet, but planned:

- Signals

Source code and issue tracking at Bitbucket.

CHAPTER

ONE

INSTALLATION

Just grab it from PyPI with *easy_install* or *pip*, for example:

```
$ easy_install Flask-Genshi
```

If you're starting a new project you don't need to explicitly install Flask as Flask-Genshi depends on it already.

CHAPTER

TWO

HOW TO USE

You need to construct a `Genshi` with your `Flask` instance.

```
from flaskext.genshi import Genshi

app = Flask(__name__)
genshi = Genshi(app)
```

The best way to render templates is to use `render_response()`. This ensures that the proper mimetype is sent if you render XHTML, XML or text, and sets the right doctype for you.

Use it like so:

```
from flaskext.genshi import render_response

@app.route('/')
def index():
    title = 'Genshi + Flask, a match made in heaven!'
    return render_response('index.html', dict(title=title))
```

CHAPTER

THREE

USING METHODS

Methods control things such as the doctype and how end tags are rendered, and with `render_response()` also the mimetype. Unless overridden the method used is decided by the template's filename extension.

By default HTML renders as strict HTML 4.01. This is how you change it to HTML5:

```
genshi.extensions['html'] = 'html5'
```

I recommend against this but of course you can also change it to XHTML:

```
genshi.extensions['html'] = 'xhtml'
```

You can also override the default with a parameter to the templating functions:

```
render_response('video.html', method='html5')
```

The extensions *html*, *xml*, *txt*, *js*, *css* and *svg* are recognized, but you can add any extension and method you like. Note that *txt*, *js* and *css* templates are rendered with `genshi.template.NewTextTemplate` which is not XML-based. Rendering javascript with templates gives you tools like `flask.url_for()` and rendering CSS with templates gives you dynamic stylesheets with things like variables.

CHAPTER

FOUR

USING JINJA FILTERS AND TESTS

Flask-Genshi supports tests and filters from Jinja:

```
<p class="${'even' if tests.even(1) else 'odd'}">
    ${filters.truncate('Hello World', 10)}
</p>
```

Result:

```
<p class="odd">
    Hello ...
</p>
```

CHAPTER

FIVE

MODULE TEMPLATES

Flask can load templates specific to a `flask.Module` and Flask-Genshi also supports this. It works just like in Flask:

```
render_response('modname/template.html')
```

This will first look for, e.g. `app/templates/modname/template.html`, and if not found, e.g. `app/mods/modname/templates/template.html`. This lets you override module templates in your application. New in version 0.4.

CHAPTER

SIX

CONTEXT PROCESSORS

Context processors work as expected.

```
@app.context_processor
def add_site_name():
    return dict(site_name=app.config['SITE_NAME'])
```

This lets you use `site_name` in templates without including it in every render call.

```
<title>$site_name</title>
```

New in version 0.4.

CHAPTER

SEVEN

RENDER FROM STRINGS

Like `flask.render_template_string()`, Flask-Genshi lets you render strings as templates:

```
render_response(string='Hello $name', context={'name': 'World'}, method='text')
```

The same pattern applies to all functions. New in version 0.4.

CHAPTER

EIGHT

API REFERENCE

class flaskext.genshi.**Genshi**(*app=None*)

Initialize extension.

```
app = Flask(__name__)
genshi = Genshi(app)
```

Changed in version 0.4: You can now initialize your application later with init_app().Deprecated since version 0.4: app.genshi_instance in favor of app.extensions['genshi'].

extensions

What method is used for an extension.

filter(**methods*)

Decorator that adds a function to apply filters to templates by rendering method.

Example:

```
from genshi.filters import Transformer

@genshi.filter('html')
def prepend_title(template):
    return template | Transformer('head/title').prepend('MySite - ')
```

See the Genshi documentation for more filters you can use. New in version 0.3.

filters

Filter functions to be applied to templates. New in version 0.3.

init_app(*app*)

Initialize a Flask application for use with this extension. Useful for the factory pattern but not needed if you passed your application to the Genshi constructor.

```
genshi = Genshi()

app = Flask(__name__)
genshi.init_app(app)
```

New in version 0.4.

methods

Render methods. Changed in version 0.3: Support for Javascript and CSS.Changed in version 0.4: Support for SVG.

template_loader

A genshi.template.TemplateLoader that loads templates from the same places as Flask.

flaskext.genshi.**generate_template**(*template=None*, *context=None*, *method=None*, *string=None*)
Creates a Genshi template stream that you can run filters and transformations on.

flaskext.genshi.**render_template**(*template=None*, *context=None*, *method=None*, *string=None*)
Renders a template to a string.

flaskext.genshi.**render_response**(*template=None*, *context=None*, *method=None*, *string=None*)
Renders a template and wraps it in a `response_class` with mimetype set according to the rendering method.

PYTHON MODULE INDEX

f