Tango Documentation

Release 0.3

Ron DuPlain

Jun 28, 2017

Contents

Pre-process web content from a variety of sources, one Python script at a time.

Tango is a web framework for content middleware, great for respinning content for mobile web sites or repurposing upstream data (no matter how messy) for new and improved APIs, built with Python and Flask.

CHAPTER 1

Overview

Here is Tango's plan:

At a glance, Tango...

- is a mobile web framework built with Python
- is a scripting layer for reflowing data... if you can pull your data via a Python script, you can serve it for free
- provides a basis for testing data integrity with unit and functional tests

Benefits

- Two teams develop Tango site packages in parallel:
- template developers, implementing designs and arranging content
- data sourcers, tapping into origin database or site to push content into templates
- Spec-first development
- Tango site developers codify site's URL routes and data using Python & yaml definitions.
- spec clearly spells out template content – develop the spec as a collection of yaml headers, then develop templates & data at the same time
- Productivity Measures
- Tango snapshots data - develop templates without fetching data
- data sourcing occurs outside of web context - develop & unit-test data modules in isolation, in a simple scripting environment

Specifics

- Tango deploys as a Python WSGI app:
- complies with the WSGI web standard.
- most deployments use mod_wsgi under Apache httpd.
- if needed, can readily port to ISAPI interface on Microsoft's IIS platform.
- Tango automates content and deployment on a schedule, including:
- automated deploy using Python standards & automated upgrade using git revision control
- dynamic views with caching – cached on a time-to-live schedule via cron
- failsafe – data updates which fail do not overwrite production data (essential in productions where an API is built from screen scraping)
- Tango site packages include
- a template package in Python's Jinja2
- a stash package in Python, using yaml headers, includes stashable content
- static assets - images, CSS, JavaScript
- config.py using simple key/value pairs
- Tango supports stand-alone Python scripts where full packages are not needed.
- Templating: Tango uses Python's highly regarded Jinja2 (inspired by Django).

Stashing Content

Stashable content is that which can be fetched up front and served to all users. In a Tango project, this content is scripted in Python modules, which have structured metadata written in yaml. When serving an application, the Tango framework walks the `sitename.stash` package or module (or accepts a single Python module for small projects), building all of the application view functions based on the yaml metadata. Simple Tango sites are just a `stash` package with a `templates` directory. A simpler Tango site is just a `stash` package with a config telling Tango to return json. The simplest Tango site is single Python module, which is treated as a `stash` and is useful in building light APIs.

Dynamic Content

Pure dynamic content and forms require custom view functions. In this case, Tango builds an `app` object from the stash module, and this `app` object allows for additional routes, view functions, and other features as provided by Flask. Projects without stashable content are effectively just Flask projects which use utilities/tools provided by Tango.

Need to drop into Flask development? Simply:

```
from tango.factory.app import build_app
app = build_app('sitename')
```

This app is a flask.Flask instance ready for any of the APIs provided by Flask, a full web framework with a small accessible core.

Other Notes

Tango:

- framework reduces web request & response code to 0.
- developers can theme sites easily using template inheritance and CSS.
- is a rapid prototyping framework (think *very* rapid), but is ready for primetime & full applications.
- provides for automated unit and functional tests, testing all the way up to (but not including) browser quirks.

On redirecting users from the desktop site:

- Most site owners target iPhone, Android, and Blackberry.
- Nearly all of these devices have JavaScript enabled.
- Use a simple JavaScript redirection script (preferably on every page, but at least the home page).
- For wider device targets:
- Set URL rewrite rules for Apache httpd or IIS.
- Redirect devices even if JavaScript is disabled.

On screen scraping:

- Sometimes the client data with the best structure is structured as (X)HTML.
- Tango does not have a general rule or silver bullet for screen scraping. Each case is treated specially. Developers study the client's markup, decide which elements to select, and strip/cleanup attributes and tags as needed. Some origin elements and attributes flow through, others are mutated. For maintenance, this requires a close eye on how the origin site changes.

Discussion Topics

On Context

Throughout the Tango project, there are two uses of the word "context":

- The Flask app current in context; here "context" is the same as used in the Flask project. (Flask has request contexts and context-locals.)
- The template context, a collection of variables available in the template; here "context" is the same as used in the Jinja project.

Logic in Templates?

Template developers say that heavy logic should stay out of templates, and there are good reasons for that. In stark contrast, Tango relies on heavy logic in the templates. This is intentional; for stashable content, *all* request-based logic is in the templates. Where Tango stashes content, there are no explicit view functions, only templates and a freestyle data layer.

Yet Another Web Framework?

No, Tango extends Flask, or rather, Tango *builds* Flask, Flask WSGI application objects to be exact. Flask:

- builds on Werkzeug, a WSGI implementation and toolkit
- builds on Jinja2, a templating platform
- allows for a Pythonic app-building pattern
- provides for extensions with clear conventions (and the Flask committers review & approve these extensions)

Tango focuses on the templating platform, completely hides the WSGI layer (but exposes APIs to WSGI if needed), establishes a spec-first development pattern on top of Flask, leverages Flask-related tools & extensions, and as a result, makes the Tango developers more productive in building mobile web sites.

Tango is WillowTree's platform on Flask, but is developed for general use.

Releases

The current release is 0.2 (Salida), released on Oct 26, 2011. All releases are guaranteed with 100% statement test coverage.

Tango is built for CPython (the reference Python implementation), for versions 2.6 and 2.7.

License

BSD.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search