
flowy Documentation

Release 0.1.5b

Sever Banesiu

May 18, 2016

1	Getting Started	3
2	Changelog	5
2.1	Next Release	5

Flowy is a workflow modeling and execution library. A workflow is a process composed of independent and inter-dependent tasks. The independent tasks can be concurrent and can run in parallel on many machines. Flowy uses single-threaded Python code to model workflows. It infers the concurrency by building the task dependency graph at run-time. A workflow engine, like Amazon's SWF, handles the task scheduling and routing. The engine also stores task results and a history of the entire workflow execution. An open source alternative to Amazon SWF is also available as part of the [Eucalyptus](#) project.

Modeling workflows with Python code is easy and familiar. It also gives the user great flexibility without sacrificing the readability. A toy example workflow, with Flowy, looks like this:

```
def sum_activity(a, b):
    time.sleep(1)
    return a + b

def square_activity(n):
    time.sleep(1)
    return n ** 2

class ExampleWorkflow(object):
    def __init__(self, square, sum):
        self.square = square
        self.sum = sum

    def __call__(self, a, b):
        a_squared = self.square(a)
        b_squared = self.square(b)
        return self.sum(a_squared, b_squared)

w = flowy.LocalWorkflow(ExampleWorkflow)
w.conf_activity('square', square_activity)
w.conf_activity('sum', sum_activity)
print(w.run(2, 10))
```

Getting Started

Flowy is available on the Python Package Index site. To install it use `pip`:

```
pip install flowy
```

Next, you should read the [Tutorial](#). It provides a narrative introduction of the most important features of Flowy. It also shows how to run a workflow on different engines.

Changelog

2.1 Next Release

- A large and backward-incompatible rewrite.
- Added a local engine that can run workflows on a single machine using threads or processes. This is handy for local development and quick prototypes.
- Added workflow execution tracing and visualization, as dot graphs, for the local engine.
- Proxy objects replaced task results. This allows a workflow to run as single threaded Python code, without Flowy. It also makes testing more convenient.
- Moved the workflow configuration outside of the workflow code. This makes it easy to configure the same workflow to run on different engines.

2.1.1 Tutorial

Subtitle1

2.1.2 Changelog

Next Release

- A large and backward-incompatible rewrite.
- Added a local engine that can run workflows on a single machine using threads or processes. This is handy for local development and quick prototypes.
- Added workflow execution tracing and visualization, as dot graphs, for the local engine.
- Proxy objects replaced task results. This allows a workflow to run as single threaded Python code, without Flowy. It also makes testing more convenient.
- Moved the workflow configuration outside of the workflow code. This makes it easy to configure the same workflow to run on different engines.