fibers Documentation

Release 1.1.0

Saúl Ibarra Corretgé

Mar 19, 2018

Contents

CHAPTER 1

Overview

Fibers are lightweight primitives for cooperative multitasking in Python. They provide means for running pieces of code that can be paused and resumed. Unlike threads, which are preemptively scheduled, fibers are scheduled cooperatively, that is, only one fiber will be running at a given point in time, and no other fiber will run until the user explicitly decides so.

When a fiber is created it will not run automatically. A fiber must be 'switched' into for it to run. Fibers can switch control to other fibers by way of the *switch* or *throw* functions, which switch control or raise and exception in the target fiber respectively.

CHAPTER 2

Motivation

I started this project mainly because I wanted a slightly different API from what greenlet offers. After playing with the code a bit I decided to make more changes and *fibers* became what it is today.

"Why not just fork greenlet and make the changes?" you probably wonder. Since I wanted to make some fundamental changes to the API changes were going to be big and since I was there I thought I may as well use the stack switching implementation used by PyPy, which has some advantages over the current one used in greenlet.

For the curious, I made a list of differences between fibers and greenlet here, see why-fibers.

CHAPTER 3

API

The `fibers` module exports two objects, the `Fiber` type and the `error` object.

class fibers.**Fiber**([*target*[, *args*[, *kwargs*[, *parent*]]]])

Parameters

- **target** (*callable*) – callable which this fiber will execute when switched to.
- **args** (*tuple*) – tuple of arguments which the specified target callable will get passed.
- **kwargs** (*dict*) – dictionary of keyword arguments the specified callable will get passed.
- **parent** (*Fiber*) – parent fiber for this object. If not specified, the current one will be used.

`Fiber` objects are lightweight microthreads which are cooperatively scheduled. Only one can run at a given time and the `switch` and/or `throw` functions must be used to switch execution from one fiber to another.

The first time a fiber is switched into, the specified callable will be called with the given arguments and keyword arguments.

```
def runner(*args, **kwargs):
    return 42

f = Fiber(target=runner, args=(1, 2, 3), kwargs={'foo': 123}
f.switch()
```

When `f.switch()` is called the `runner` function will be executed, with the given positional and keyword arguments.

switch([*value*])

Parameters **value** (*object*) – Arbitrary object which will be returned to the fiber which called `switch` on this fiber before. A value can only specified if the fiber has already been started. In order to start a fiber, this function must be called without a value.

Suspend the current running fiber and switch execution to the target fiber. If a value is specified, the fiber which previously called `switch()` will appear to return the value specified here.

throw(*typ*[, *val*[, *tb*]])

Parameters

- **typ** (*type*) – Exception type to be raised.
- **val** (*object*) – Exception instance value to be raised.
- **tb** (*traceback*) – Traceback object.

Suspend the current running fiber and switch execution to the target fiber, raising the specified exception immediately. The fiber which is resumed will get the exception raised, and if it's not caught it will be propagated to the parent.

is_alive()
Returns *True* if the fiber hasn't ended yet, *False* if it has already ended.

classmethod current()
Returns the current `Fiber` object.

exception fibers.**error**
Exception raised by this module when an error such as trying to switch to a fiber in a different thread occurs.

fibers.**current**()
Returns the current `Fiber` object.

CHAPTER 4

Parents

Fibers are organized in a tree form. Each native Python thread has a fibers tree, which is initialized the first time a fiber is created. When a fiber is created the user can select what the parent fiber will be. When that fiber finishes execution, control will be switched to the parent.

CHAPTER 5

Multi-threading

There is no multithreading support, that is, a fiber in one thread cannot switch control to a fiber in a different thread, this will raise an exception. Likewise, a fiber cannot get assigned a parent which belongs to a different thread.

Note: a fiber is bound to the thread where it was created, and this cannot be changed.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index