appendonly Documentation

Release 1.2

Tres Seaver

December 28, 2014

Contents

This package provides a set of data structures for use in ZODB applications where standard BTrees are poor fits for an application's requirements.

In particular, these data structures are designed to minimize conflict errors when doing frequent "append" operations to queues and stacks.

CHAPTER 1

AppendStack

This class provides a LIFO stack of separately-persisted objects:

- The stack manages a set of "layer" objects, with a configurable limit on the number of layers. Each layer has a configurable maximum length, and a sequential generation number.
- The stack appends items to most recent layer until the layer is filled; it then adds a new layer.
- If the number of layers then exceeds the configured maximum, the stack prunes the oldest layer(s) to conform to that limit.
- When pruning, the stack calls an application-supplied callback for archiving / cleanup of the pruned layer.
- Iteration over the stack yields (generation, index, object) tuples. in reverse order to that in which the objects were appended.

The stack is implemented as a single persistent record, with custom ZODB conflict resolution code.

Archive

This class provides a linked list of separately-persisted copies of layer data pruned from an `AppendStack``. The intended use would be something like:

```
from appendonly import AppendStack
from appendonly import Archive

class RecentItems(object):

    def __init__(self):
        self._recent = AppendStack()
        self._archive = Archive()

    def pushItem(self, object):
        self._recent.push(object, self._archive.addLayer)

    def __iter__(self):
        for generation, index, item in self._recent:
            yield item
        for generation, index, item in self._archive:
            yield item
```

CHAPTER 3

Accumulator

This class provides a list-like data structure, where the only mutations allowed are to append to or clear the list. Intended uses are for a set of pending operations / changes / notifications, which get processed as a unit (at which point the accumulator is cleared).