



repoze.folder Documentation

Release 0.5

Repoze Developers

December 28, 2014

1	Using repoze.folder	3
2	API Documentation for repoze.folder	5
2.1	Folder interface	5
2.2	Events sent when <code>Folder.__setitem__</code> called	5
2.3	Events sent when <code>Folder.__delitem__</code> called	5
3	repoze.folder Change History	7
4	Changes	9
4.1	1.0 (2014-12-28)	9
4.2	0.6.3 (2012-03-29)	9
4.3	0.6.2 (2010-10-04)	9
4.4	0.6.1 (2010-10-01)	9
4.5	0.6 (2010-09-30)	9
4.6	0.5 (2010/09/04)	10
4.7	0.4 (2009/06/15)	10
4.8	0.3.5 (2009/1/8)	10
4.9	0.3.4 (2009/1/8)	10
4.10	0.3.3 (2009/1/6)	10
4.11	0.3.2 (2008/12/13)	11
4.12	0.3.1 (2008/12/13)	11
4.13	0.3 (2008/12/13)	11
4.14	0.2.1 (2008/10/31)	11
4.15	0.2 (2008/10/22)	11
4.16	0.1 (2008/10/13)	11
5	Indices and tables	13

`repoze.folder` provides a barebones ZODB `folder` (container) implementation with object event support.

Using repoze.folder

repoze.folder provides a barebones folder implementation with object event support. Folders have a dictionary-like interface and emit “object events” on the addition and removal of objects when certain methods of this interface are exercised.

Using a folder:

```

1  >>> from repoze.folder import Folder
2  >>> from persistent import Persistent
3  >>> folder = Folder()
4  >>> class Child(Persistent):
5  >>>     pass
6  >>> folder['child1'] = Child()
7  >>> folder['child2'] = Child()
8  >>> list(folder.keys())
9  ['child1', 'child2']
10 >>> folder.get('child1')
11 <Child object at ELIDED>
12 >>> del folder['child1']
13 >>> list(folder.keys())
14 ['child2']

```

Folder objects are based on BTree code, so as long as you persist them, the folder should be able to contain many objects efficiently.

To subscribe to object events that occur when a folder’s `__setitem__` or `__delitem__` is called, you can place ZCML in your application’s registry to handle the events:

```

<subscriber for=".interfaces.IChild
                repoze.folder.interfaces.IObjectAddedEvent"
                handler=".subscribers.child_added"/>

```

The event interface types are as follows:

```

IObjectWillBeAddedEvent (before an object is seated into the folder)
IObjectAddedEvent (after the object is seated into the folder)
IObjectWillBeRemovedEvent (before the object is removed from the folder)
IObjectRemovedEvent (after the object is removed from the folder)

```

See the `repoze.folder.interfaces` file for more information about the folder API and the event object APIs.

API Documentation for repoze.folder

2.1 Folder interface

interface `repoze.folder.interfaces.IFolder`

A Folder which stores objects using Unicode keys.

All methods which accept a name argument expect the name to either be Unicode or a byte string decodable using the default system encoding or the UTF-8 encoding.

2.2 Events sent when `Folder.__setitem__` called

interface `repoze.folder.interfaces.IObjectWillBeAddedEvent`

Extends: `zope.interface.interfaces.IObjectEvent`

An event type sent when an before an object is added

interface `repoze.folder.interfaces.IObjectAddedEvent`

Extends: `zope.interface.interfaces.IObjectEvent`

An event type sent when an object is added

2.3 Events sent when `Folder.__delitem__` called

interface `repoze.folder.interfaces.IObjectWillBeRemovedEvent`

Extends: `zope.interface.interfaces.IObjectEvent`

An event type sent before an object is removed

interface `repoze.folder.interfaces.IObjectRemovedEvent`

Extends: `zope.interface.interfaces.IObjectEvent`

An event type sent when an object is removed

`repoze.folder` Change History

Changes

4.1 1.0 (2014-12-28)

- Add support for PyPy.
- Add support for Python 3.2, 3.3, and 3.4.
- Add support for testing on Travis.
- Drop support for Python 2.4 and 2.5.

4.2 0.6.3 (2012-03-29)

Note: This release is the last which will maintain support for Python 2.4 / Python 2.5.

- Add support for continuous integration using `tox` and `jenkins`.
- Add ‘`setup.py dev`’ alias (runs `setup.py develop` plus installs `nose` and `coverage`).
- Move to GitHub.

4.3 0.6.2 (2010-10-04)

- Fix iteration bug due to use of `_order` as tuple.

4.4 0.6.1 (2010-10-01)

- Fixed persistence bugs in ordering support when adding or removing items.

4.5 0.6 (2010-09-30)

- Add support for ordering items in a folder.

4.6 0.5 (2010/09/04)

- Make `remove` return the removed object.
- Add `pop` method.

4.7 0.4 (2009/06/15)

- 100% test coverage.
- Add an `add` method that does what `__setitem__` does. It also provides a flag named `send_events`, which by default is `True`. If it is `False` when `add` is called, folder events (`IOObjectWillBeAddedEvent` and `IOObjectAddedEvent`) will not be sent.
- Add a `remove` method that does what `__delitem__` does. It also provides a flag named `send_events`, which by default is `True`. If it is `False` when `add` is called, folder events (`IOObjectWillBeRemovedEvent` and `IOObjectRemovedEvent`) will not be sent.

4.8 0.3.5 (2009/1/8)

- Add a `BTrees.Length` object to folders that don't already have one during `__setitem__` and `__delitem__` (this is an “evolution” step; having a `Length` object is useful for performance reasons).

4.9 0.3.4 (2009/1/8)

- Fix backwards compatibility foul (near `self._num_objects.change(1): AttributeError: 'NoneType' object has no attribute 'change'`).

4.10 0.3.3 (2009/1/6)

- Add tests for `unicodify` and make docs about to-Unicode convenience conversion from byte strings (and error messages) slightly clearer.
- Now no matter what is passed to the folder as constructor, we try to turn it into an `OBTTree` (before it was set as data on the instance without any conversion).
- A `__len__` method was added to `repoze.folder.Folder` instances. It returns the number of subobjects in the folder.
- A `_num_objects` attribute is set onto newly created `repoze.folder.Folder` instances. This is a `BTrees.Length.Length` object. We manage this length object in order to supply a return value for the `__len__` method instead of using the folder's underlying `OBTTree.__len__` method (querying a btree for length can be arbitrarily expensive). A `_num_objects` class attribute was added equalling `None` to provide a backward compatibility cue for already-persisted objects which do not have a meaningful `Length` attribute.
- The implementation no longer concerns itself with advertising a modified event (`IOObjectModifiedEvent`).

4.11 0.3.2 (2008/12/13)

- Yeah. 0.3.1 was another brownbag, as we need to try to decode ASCII to unicode before we use the utf-8 decoding.

4.12 0.3.1 (2008/12/13)

- Mistakenly removed `__parent__` and `__name__` attributes from folder implementation, making 0.3 a brownbag.

4.13 0.3 (2008/12/13)

4.13.1 Backwards Incompatibilities

- When a new object is added using `__setitem__` with the same name as an existing object, a `KeyError` is now raised rather than the item being silently replaced.
- API methods accepting a name (`__setitem__`, `__getitem__`, `get`, `__contains__`, and `__delitem__`) now attempt to decode bytestrings to Unicode using the utf-8 encoding before performing the action the method implies.
- Previously, it was possible to store either an ASCII bytestring or a Unicode object as a key value. Now all key values are converted to Unicode before being stored.

4.14 0.2.1 (2008/10/31)

- Remove `__init__` from IFolder interface.

4.15 0.2 (2008/10/22)

- Update Sphinx docs, using interfaces
- Add folder `__name__` to repr and str of folder in output.

4.16 0.1 (2008/10/13)

- Initial release.

Indices and tables

- *genindex*
- *modindex*
- *search*

I

IFolder (interface in `repoze.folder.interfaces`), [5](#)
IObjecAddedEvent (interface in `repoze.folder.interfaces`), [5](#)
IObjecRemovedEvent (interface in `repoze.folder.interfaces`), [5](#)
IObjecWillBeAddedEvent (interface in `repoze.folder.interfaces`), [5](#)
IObjecWillBeRemovedEvent (interface in `repoze.folder.interfaces`), [5](#)