zodburi Documentation

Release 2.6.0

Repoze Developers

May 17, 2023

Contents

CHAPTER 1

Overview

A library which parses URIs and converts them to ZODB storage objects and database arguments.

It will run under CPython 3.7+ and pypy3. It will not run under Jython. It requires ZODB >= 5.0.0.

CHAPTER 2

Installation

Install using pip, e.g. (within a virtualenv):

```
$ pip install zodburi
```

CHAPTER 3

Using

`zodburi` has exactly one api: *`zodburi.resolve_uri()`*. This API obtains a ZODB storage factory and a set of keyword arguments suitable for passing to the `ZODB.DB.DB` constructor. For example:

```
from zodburi import resolve_uri

storage_factory, dbkw = resolve_uri(
                   'zeo://localhost:9001?connection_cache_size=20000')

# factory will be an instance of ClientStorageURIResolver
# dbkw will be {'connection_cache_size':20000, 'pool_size':7,
#               'database_name':'unnamed'}

from ZODB.DB import DB
storage = storage_factory()
db = DB(storage, **dbkw)
```

CHAPTER 4

URI Schemes

The URI schemes currently recognized in the `zodbconn.uri` setting are `file://`, `zeo://`, `zconfig://`, `memory://` and `demo:`. Documentation for these URI scheme syntaxes are below.

In addition to those schemes, the relstorage package adds support for `postgres://`.

4.1 `file://` URI scheme

The `file://` URI scheme can be passed as `zodbconn.uri` to create a ZODB FileStorage database factory. The path info section of this scheme should point at a filesystem file path that should contain the filestorage data. For example:

```
file:///my/absolute/path/to/Data.fs
```

The URI scheme also accepts query string arguments. The query string arguments honored by this scheme are as follows.

4.1.1 FileStorage constructor related

These arguments generally inform the FileStorage constructor about values of the same names.

create boolean

read_only boolean

quota bytesize

4.1.2 Database-related

These arguments relate to the database (as opposed to storage) settings.

database_name string

4.1.3 Connection-related

These arguments relate to connections created from the database.

connection_cache_size integer (default 10000) target size, in number of objects, of each connection's object cache

connection_cache_size_bytes integer (default 0) target estimated size, in bytes, of each connection's object cache

0 means no limit.

A suffix of KB, MB, or GB may be used to provide units.

connection_historical_cache_size integer (default 1000) target size, in number of objects, of each historical connection's object cache

connection_historical_cache_size_bytes integer (default 0) target estimated size, in bytes, of each historical connection's object cache

0 means no limit.

A suffix of KB, MB, or GB may be used to provide units.

connection_historical_pool_size integer (default 3) expected maximum total number of historical connections simultaneously open

connection_historical_timeout integer (default 300) maximum age of inactive historical connections

When a historical connection has remained unused in a historical connection pool for more than connection_historical_timeout seconds, it will be discarded and its resources released.

connection_large_record_size integer (default 16MB) record size limit before suggesting using blobs

When object records are saved that are larger than this, a warning is issued, suggesting that blobs should be used instead.

A suffix of KB, MB, or GB may be used to provide units.

connection_pool_size integer (default 7) expected maximum number of simultaneously open connections

There is no hard limit (as many connections as are requested will be opened, until system resources are exhausted). Exceeding pool-size connections causes a warning message to be logged, and exceeding twice pool-size connections causes a critical message to be logged.

connection_pool_timeout integer (default unlimited) maximum age of inactive (non-historical) connections

When a connection has remained unused in a connection pool for more than connection_pool_timeout seconds, it will be discarded and its resources released.

4.1.4 Blob-related

If these arguments exist, they control the blob settings for this storage.

blobstorage_dir string

blobstorage_layout string

4.1.5 Misc

demostorage boolean (if true, wrap FileStorage in a DemoStorage)

4.1.6 Example

An example that combines a path with a query string:

```
file:///my/Data.fs?connection_cache_size=100&blobstorage_dir=/foo/bar
```

4.2 zeo:// URI scheme

The `zeo://` URI scheme can be passed as `zodbconn.uri` to create a ZODB ClientStorage database factory. Either the host and port parts of this scheme should point at a hostname/portnumber combination e.g.:

```
zeo://localhost:7899
```

Or the path part should point at a UNIX socket name:

```
zeo:///path/to/zeo.sock
```

The URI scheme also accepts query string arguments. The query string arguments honored by this scheme are as follows.

4.2.1 ClientStorage-constructor related

These arguments generally inform the ClientStorage constructor about values of the same names.

storage string

cache_size bytesize

name string

client string

debug boolean

var string

min_disconnect_poll integer

max_disconnect_poll integer

wait_for_server_on_startup (deprecated alias for wait) boolean

wait boolean

wait_timeout integer

read_only boolean

read_only_fallback boolean

drop_cache_rather_verify boolean

username string

password string

realm string

blob_dir string

shared_blob_dir boolean

blob_cache_size bytesize

blob_cache_size_check integer

client_label string

4.2.2 Misc

demostorage boolean (if true, wrap ClientStorage in a DemoStorage)

4.2.3 Connection-related

These arguments relate to connections created from the database.

connection_cache_size integer (default 10000)

connection_pool_size integer (default 7)

4.2.4 Database-related

These arguments relate to the database (as opposed to storage) settings.

database_name string

4.2.5 Example

An example that combines a path with a query string:

```
zeo://localhost:9001?connection_cache_size=20000
```

4.3 `zconfig://` URI scheme

The `zconfig://` URI scheme can be passed as `zodbconn.uri` to create any kind of storage that ZODB can load via ZConfig. The path info section of this scheme should point at a ZConfig file on the filesystem. Use an optional fragment identifier to specify which database to open. This URI scheme does not use query string parameters.

4.3.1 Examples

An example ZConfig file:

```
<zodb>
  <mappingstorage>
  </mappingstorage>
</zodb>
```

If that configuration file is located at /etc/myapp/zodb.conf, use the following URI to open the database:

```
zconfig:///etc/myapp/zodb.conf
```

A ZConfig file can specify more than one database. For example:

```
<zodb temp1>
  <mappingstorage>
  </mappingstorage>
</zodb>
<zodb temp2>
  <mappingstorage>
  </mappingstorage>
</zodb>
```

In that case, use a URI with a fragment identifier:

```
zconfig:///etc/myapp/zodb.conf#temp1
```

4.4 `memory://` URI scheme

The `memory://` URI scheme can be passed as `zodbconn.uri` to create a ZODB MappingStorage (memory-based) database factory. The path info section of this scheme should be a storage name. For example:

```
memory://storagename
```

However, the storage name is usually omitted, and the most common form is:

```
memory://
```

The URI scheme also accepts query string arguments. The query string arguments honored by this scheme are as follows.

4.4.1 Database-related

These arguments relate to the database (as opposed to storage) settings.

database_name string

4.4.2 Connection-related

These arguments relate to connections created from the database.

connection_cache_size integer (default 10000)

connection_pool_size integer (default 7)

4.4.3 Example

An example that combines a dbname with a query string:

```
memory://storagename?connection_cache_size=100&database_name=fleeb
```

4.5 demo: URI scheme

The `demo:` URI scheme can be passed as `zodbconn.uri` to create a DemoStorage database factory. DemoStorage provides an overlay combining base and δ ("delta", or in other words, "changes") storages. The URI scheme contains two parts, base and δ:

```
demo:(base_uri)/(δ_uri)
```

an optional fragment specifies arguments for `ZODB.DB.DB` constructor:

```
demo:(base_uri)/(δ_uri)#dbkw
```

4.5.1 Example

An example that combines ZEO with local FileStorage for changes:

```
demo:(zeo://localhost:9001?storage=abc)/(file:///path/to/Changes.fs)
```

CHAPTER 5

More Information

5.1 `zodburi` API

zodburi.**resolve_uri**(*uri*)

Returns a tuple, (factory, dbkw) where factory is a no-arg callable which returns a storage matching the spec defined in the uri. dbkw is a dict of keyword arguments that may be passed to ZODB.DB.DB.

CHAPTER 6

Reporting Bugs / Development Versions

Visit https://github.com/Pylons/zodburi to download development or tagged versions.

Visit https://github.com/Pylons/zodburi/issues to report bugs.

CHAPTER 7

Change Log

7.1 2.6.0 (2023-05-17)

- Stop support for ZODB4
- Stop support for python<3.7

7.2 2.5.0 (2021-05-12)

- Support both ZODB4 and ZODB5.
- Add support for PyPy.
- Add support for Python 3.8.
- Drop support for Python 3.4.
- Add support for `demo:` URI scheme.

7.3 2.4.0 (2019-01-11)

- Add support for Python 3.7.
- Fix PendingDeprecationWarning about `cgi.parse_qsl`. (PR #21)

7.4 2.3.0 (2017-10-17)

- Fix parsing of `zeo://` URI with IPv6 address.
- Drop support for Python 3.3.

- Add support for Python 3.6.

7.5 2.2.2 (2017-05-05)

- Fix transposed `install_requires` and `tests_require` lists in `setup.py`.

7.6 2.2.1 (2017-04-18)

- Fix breakage added in 2.2 to the `zconfig` resolver.

7.7 2.2 (2017-04-17)

- Add support for additional database configuration parameters: `pool_timeout`, `cache_size_bytes`, `historical_pool_size`, `historical_cache_size`, `historical_cache_size_bytes`, `historical_timeout`, and `large_record_size`.

7.8 2.1 (2017-04-17)

- Add support for Python 3.4 and 3.5.
- Drop support for Python 2.6 and 3.2.
- Add missing ClientStorage constructor kw args to resolver.

7.9 2.0 (2014-01-05)

- Update `ZODB3` meta-package dependency to `ZODB` + `ZConfig` + `ZEO`. Those releases are what we import, and have final Py3k-compatible releases.
- Packaging: fix missing `url` argument to `setup()`.

7.10 2.0b1 (2013-05-02)

- Add support for Python 3.2 / 3.3.
- Add `setup.py docs` alias (runs `setup.py develop` and installs documentation dependencies).
- Add `setup.py dev` alias (runs `setup.py develop` and installs testing dependencies).
- Automate building the Sphinx docs via `tox`.
- Fix `zconfig:` URIs under Python 2.7. The code worked around a bug in the stdlib's `urlparse.urlsplit` for Python < 2.7; that workaround broke under 2.7. See https://github.com/Pylons/zodburi/issues/5
- Drop support for Python 2.5.

7.11 1.1 (2012-09-12)

- Remove support for `postgres://` URIs, which will now be provided by the `relstorage` package. Thanks to Georges Dubus for the patch!

7.12 1.0 (2012-06-07)

- Add support for `postgres://` URIs. Thanks to Georges Dubus for the patch!
- Pin dependencies to Python 2.5-compatible versions when testing with tox under Python 2.5.
- Update the documentation for publication to ReadTheDocs

7.13 1.0b1 (2011-08-21)

- Initial release.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

z

Index

R

Z