msaStorageDict Module

`.zookeeper`

Classes

MSAZookeeperDict

Bases: `MSAConnectionStorageDict`

Dictionary backed by Zookeeper. Functions just as you would expect a normal dictiony to function, except the values in the dictionary are persisted and loaed from Zookeper located at the spcified `path` in the constructor.

Due to Zookeeper's watchers, this dictionary has the nice property that the called to its `last_updated` method (to check if the storage has been updated since the dict was last synced) simply returns a cached value -- it does not query Zookeeper or anything like that, so it's basically free. This means that you can run this dictionary with `autosync=True` and it will still be reasonably performant.

Dictionary keys in a `MSAZookeeperDict` are stored at inividual nodes in the zookeeper heirarchy, with the value of the node being the value of that key. Each node for each dict key is a child of the "root" node, whose path is specified with the `path` argument in the constructor.

```
    from storagedict import MSAZookeeperDict
    from kazoo.client import KazooClient
    kazoo = KazooClient()
    kazoo.start()
    zkdict = MSAZookeeperDict(kazoo, '/app/config')
    zkdict['exchange_rate'] = 25
    zkdict['language'] = 'en-US'
    zkdict['exchange_rate']
    # 25
    zkdict['language']
    # 'en-US'
    zkdict.pop('exchange_rate')
    # 25
    zkdict['exchange_rate']

    # Traceback ... KeyError: 'exchange_rate'
```

NOTE: unlike `MSARedisDict` or `MSAStorageDict`, which are backed by hightly consistent backend storages, `MSAZookeeperDict` is backed with Zookeeper, which has looser consistency guarantees.

Please see this page in the Zookeeper docs for details:

http://zookeeper.apache.org/doc/r3.1.2/zookeeperProgrammers.html#ch_zkGuarantees

The basic things to keep in mind are

Sequential Consistency: Updates from a client will be applied in the order that they were sent.

Atomicity: Updates either succeed or fail -- there are no partial results.

Single System Image: A client will see the same view of the service regardless of the server that it connects to.

Reliability: Once an update has been applied, it will persist from that time forward until a client overwrites the update. This guarantee has two corollaries:

```
If a client gets a successful return code, the update will have been
applied. On some failures (communication errors, timeouts, etc) the
client will not know if the update has applied or not. We take steps
to minimize the failures, but the only guarantee is only present
with successful return codes. (This is called the monotonicity
condition in Paxos.)

Any updates that are seen by the client, through a read request or
successful update, will never be rolled back when recovering from
server failures.
```

Timeliness: The clients view of the system is guaranteed to be up-to-date within a certain time bound. (On the order of tens of seconds.) Either system changes will be seen by a client within this bound, or the client will detect a service outage.

The cliff notes version of this is that a client's view of the world will always be consistent (you can read your own writes), but updates from other clients can take time to propogate to other clients.

Functions

__increment_last_updated

```
__increment_last_updated(children = None)
```

__init__

```
__init__(*args, **kwargs)
```

Construct a new instance of a `MSAZookeeperDict`.

PARAMETER	DESCRIPTION
`connection`	Zookeeper client, likely `KazooClient`
`keyspace`	str, The path to the root config node.
`autosync`	bool, Sync with Zookeeper before each read.

__inner_set_default

```
__inner_set_default(key, value)
```

> Tries to return the value at key. If the key does not exist, attempts to create it with the value. If the node is created in the mean time, a `NodeExistsError` will be raised.

__path_of

```
__path_of(key)
```

__set_or_create

```
__set_or_create(key, value)
```

connection_hook

```
connection_hook()
```

depersist

```
depersist(key: str)
```

> Remove `key` from dictionary.

PARAMETER	DESCRIPTION
`key`	str, Key to remove from Zookeeper. **TYPE:** `str`

durables

```
durables()
```

> Dictionary of all keys and their values in Zookeeper.

last_updated

```
last_updated()
```

> Ever-increasing integer, which is bumped any time a key in Zookeeper has been changed (created, updated, deleted).
>
> The value in incremented manually by an instances when updating the dict, as well as when other instances of the dict update persistant storage, via a Zookeeper watch on the root config node.

no_node_error property

```
no_node_error()
```

persist

```
persist(key: str, value: Any)
```

> Encode and save `value` at `key`.

PARAMETER	DESCRIPTION
`key`	str, Key to store `value` at in Zookeeper. **TYPE:** `str`
`value`	Value to store. Encoded before being stored. **TYPE:** `Any`

Functions

validate_key

```
validate_key(func)
```

> Decorator to validate a key for zookeeper.

Last update: September 24, 2022

Created: September 24, 2022