google.appengine.ext.testbed package

Summary

A module to use service stubs for testing.

To test applications which use App Engine services such as the datastore, developers can use the available stub implementations. Service stubs behave like the original service without permanent side effects. The datastore stub for example allows to write entities into memory without storing them to the actual datastore. This module makes using those stubs for testing easier.

Here is a basic example: '" import unittest

from google.appengine.ext import db from google.appengine.ext import testbed

class TestModel(db.Model):

    number = db.IntegerProperty(default=42)

class MyTestCase(unittest.TestCase):

def setUp(self):

Enable the capability stub.

Parameters

enable -- True, if the fake service should be enabled, False if real service should be disabled.

init_channel_stub(enable=True)

Enable the channel stub.

Parameters

enable -- True, if the fake service should be enabled, False if real service should be disabled.

init_datastore_v3_stub(enable=True, datastore_file=None, use_sqlite=False, auto_id_policy='sequential', **stub_kw_args)

Enable the datastore stub.

The 'datastore_file' argument can be the path to an existing datastore file, or None (default) to use an in-memory datastore that is initially empty. If you use the sqlite stub and have 'datastore_file' defined, changes you apply in a test will be written to the file. If you use the default datastore stub, changes are _not_ saved to disk unless you set save_changes=True.

Note that you can only access those entities of the datastore file which have the same application ID associated with them as the test run. You can change the application ID for a test with setup_env().

Parameters

- enable -- True if the fake service should be enabled, False if real service should be disabled.
- datastore_file -- Filename of a dev_appserver datastore file.

google.appengine.ext.testbed package

Summary

A module to use service stubs for testing.

To test applications that use App Engine services, such as datastore, developers can use the available stub implementations. Service stubs behave like the original service without causing permanent side effects. The datastore stub, for example, allows you to write entities into memory without storing them to the actual datastore. The testbed module makes using those stubs for testing easier.

Example:

```
import unittest

from google.appengine.ext import db
from google.appengine.ext import testbed


class TestModel(db.Model):
  number = db.IntegerProperty(default=42)
```

```
class MyTestCase(unittest.TestCase):

  def setUp(self):
    # First, create an instance of the Testbed class.
    self.testbed = testbed.Testbed()
    # Then activate the testbed, which will allow you to use
    # service stubs.
    self.testbed.activate()
    # Next, declare which service stubs you want to use.
    self.testbed.init_datastore_v3_stub()
    self.testbed.init_memcache_stub()

  def tearDown(self):
    # Don't forget to deactivate the testbed after the tests are
    # completed. If the testbed is not deactivated, the original
    # stubs will not be restored.
    self.testbed.deactivate()

  def testInsertEntity(self):
    # Because we use the datastore stub, this put() does not have
    # permanent side effects.
    TestModel().put()
    fetched_entities = TestModel.all().fetch(2)
    self.assertEqual(1, len(fetched_entities))
    self.assertEqual(42, fetched_entities[0].number)
```

Enable stubs and disable services

The testbed module allows you to use stubs for the following services:

- capability_service
- channel

init_channel_stub(enable=True)

Enables the channel stub.

Parameters

enable -- `True` if the fake service should be enabled, or `False` if the real service should be disabled.

init_datastore_v3_stub(enable=True, datastore_file=None, use_sqlite=False, auto_id_policy='sequential', **stub_kw_args)

Enables the datastore stub.

The `datastore_file` argument can be set to the path of an existing datastore file, or `None` (default) to use an in-memory datastore that is initially empty. If you use the sqlite stub and have defined `datastore_file`, the changes that you apply in a test will be written to the file. If you use the default datastore stub, changes are not saved to disk unless you set `save_changes=True`.

Note:

You can only access those entities of the datastore file that use the same application ID as the test run. You can change the application ID for a test with `setup_env()`.

Parameters

- enable -- `True` if the fake service should be enabled, or `False` if the real service should be disabled.
- datastore_file -- File name of a dev_appserver datastore file.
- use_sqlite -- `True` to use the Sqlite stub, or `False` (default) to use the file stub.