tuxmodule

Tuxedo module for Python

Welcome to the Python Module for Oracle Tuxedo (tuxmodule) Project Homepage

This module provides access to the Tuxedo ATMI API for the Python programming language.

Features

- Write Tuxedo clients (both /WS and native) in Python (2.x and 3.2)
- Write Tuxedo servers
- Dynamic reload of python code for servers (optional)
- Support for STRING and FML buffer types. FML buffers are mapped to Python dictionaries with list elements.
- Full Tuxedo ATMI support (tpcall, tpacall, tpgetrply, tppost, tpsubscribe, tpnotify, tpsetunsol, tpdequeue, tpenqueue, tpbegin, tpabort, tpcommit, tpsectxt, tpinit, tpadvertise, userlog, tpsetctxt (multithreaded client for Tuxedo > Rel 7))

Build and Install

First, build the module:

- Check-out or download the source code from the Downloads or Source tab.
- Set TUXDIR (for example, `export TUXDIR=/opt/bea/tuxedo8.1`)
- Make sure that you use the desired version of python on the command line
- Run `python setup.py build --force` to compile the sources
- Run `python setup.py install` to install in `YOUR_PYTHON_LIB_DIR/site-packages/tuxedo`
- A package `tuxedo` will be built, with the shared objects atmi.so and atmiws.so in it.

To run the example / test:

- `cd test`
- The IPC key is set to 77662 and the WSH port is 7766. Adjust in `ubbconfig` and `setenv` if you need other values.
- Source setenv (`. ./setenv`) - make sure you still have `TUXDIR` set!
- Run `make` to build `TUXCONFIG`, `QFS` and the executables for servers and clients.
- Run `tmboot -y` to start the Tuxedo application.
- Run `testclient.py` - this will test the ATMI interface.
- Run `send.py` to test a conversational service.
- Run `simpcl.py`, the equivalent of `$TUXDIR/apps/simpapp`.
- Run `tmshutdown -y` followed by `make clean` when you are done. This also cleans up Tuxedo's IPC resources.

Usage

Remote (/WS) and local client versions of the library will be built. Use the following for the (local) native client:

```
from tuxedo.atmi import *
```

and this one for the (network remote) /WS client:

```
from tuxedo.atmiws import *
```

"simpapp" written in Python

The Tuxedo installation contains a simple example application, "simpapp". This is the Python version of that application:

simpcl.py:

```
import tuxedo.atmi

print tuxedo.atmi.tpcall("TOUPPER", "Hello World")
```

simpserv.py:

```
from tuxedo.atmi import *
import string
import sys

class server:
   def init(self, arg):
      tpadvertise("TOUPPER")

   def TOUPPER(self, arg):
      userlog("Client-ID = %s" % (self.cltid))
      return string.upper(arg)


if __name__ == '__main__':
   tuxedo.atmi.mainloop(sys.argv, server(), None)
```

For information on how to use all the ATMI functions that are implemented by the module, have a look at the examples in the `testclient.py` and `pyserver.py` files in the `test` directory.