PyCoral,
Python interface to CORAL

By

S.S.Tomar

RRCAT, Indore, India

What is it?

- PyCoral is an extension module of python, developed using the python/C API.
- It is a python interface to the CORAL package.
- PyCoral module implements python wrappers around a subset of CORAL specific C++ classes.
- It is one of the project in the framework of collaboration between RRCAT & CERN-IT-PSS/ LCG.

Why to use it?

- Write simple python scripts instead of complex C++ programs, for persistent storage/retrieval of your data in relational domain (***Oracle, MySql, SQLite, Frontier***)

Whom is it aimed at?

- *“Command line administrators”* for building CORAL API based tools using python scripts. For eg: the “Copy database tool” of the 3D project.
- Developers of Other tools of the *3D (Distributed Deployment of Databases) project*.

How to use it?

- Start Python Interpreter.
- Example 1:

```
import coral
list1 = coral.AttributeList()
            OR
from coral import *
list1 = AttributeList()
```

Examples

- Example 2:

```
import cPickle
import coral
w = coral.AttributeList()
w.extend("h","blob")
li1 = []
li2 = []
for j in range(1,500):
 li1.append(j)
cPickle.dump(li1,h,1)
li2 = cPickle.load(w[0].data())
              OR
li2 = cPickle.loads(w[0].data().readline())
```

Examples

- Example 3:

```
cursor = query.execute()
 while (cursor.next()):       // C++ style
  currentRow = cursor.currentRow()
  print str(currentRow)
          OR
  for currentRow in cursor: // Python style
   print str(currentRow)
         OR
 for currentRow in query.execute():
  print str(currentRow)
```

Examples

- Example 4:
- cmp(list1,list2)
- print str(attList1)
- attrList[0].setData(None)
- attrList['X'].setData(None)
- print attrList[0].data()
- for attribute in attrList:

  print attribute.data()

Implementation Details

- Choice of Python/C API
- Selection of the subset of CORAL classes
- Factoring of Code
- Exposed and Unexposed classes
- Exception Handling
- Parent Child relationship
- BLOB implementation
- “Attribute in AttributeList” implementation
- Inheritance implementation
- Testing techniques

Choice of Python/C API

- Boost::Python considered.
- Advantages of Python/C API
  - No external dependency except python client libraries.
  - Python style and semantics is easier to implement.
  - Although more coding but straightforward one, mostly related to struct initializations
  - No change required in the underlying C++ code.

Python style & Semantics

- For Templated methods in CORAL, Type checking is performed in the wrapper methods, which preserves the loosely-typed semantics of Python.
- Blob, implemented using the buffer interface feature in python.
- Pickling support for writing and reading python objects into BLOB.

…(Contd)

- AttributeList & Cursor with iterator protocol:

```
for attribute in attributeList:
 print attribute.data()
```

- Comparison of two Attributes or AttributeLists using “cmp” command.
- String representation of Attribute or AttributeList using “str” command.

Selection of a subset of CORAL classes

- All CoralBase/RelationalAccess except:
  - Exception classes.
  - Developer level interfaces (IRelationalService, ISession, IAuthenticationService, ILookupService etc…)

…Contd

- No 1-1 mapping of methods, to maintain python style (*for eg: templated methods, size & toOutputStream not implemented as is*)

Factoring of Code

- Based on code classification
  - Module Naming and Initialization part of the code
  - Various structure definitions like, class related PyTypeObject structures, normal method structures, mapping methods & buffer structure part of code.
  - Init and dealloc methods of the classes.
  - Various other method related code as required by specific classes in the interfaces.
- Exposed & Unexposed Classes
  - Exposed classes, header files in PyCoral subdir.
  - Unexposed classes & other code in src subdir.

Exposed & Unexposed classes

- Exposed ones can be used by other extension modules,
- Can be instantiated by the python programmer,
- Are *AttributeList, Date, Blob, TimeStamp, TableDescription, Context, ConnectionService & Exception.*
- Unexposed class objects can only be created using some functions in the exposed classes & already created unexposed class objects.

Exception Handling

- All C++ exceptions generated by CORAL classes and methods are caught & thrown in the methods of the PyCoral and can be caught further in the python code.
- Wrappers not created for Exception classes and its hierarchy in CORAL package.

Parent Child relationship

- Must to implement, because of exposed & unexposed class implementation.
- Helps in keeping track of the chain of objects created using the methods of the exposed classes.
- Tracking required for performing “dealloc” of objects in reverse order when the object that created it goes out of scope.

BLOB implementation

- By using the buffer protocol, which allows its reading and writing as a buffer without additional memory requirement.
- Pickling support, for reading and writing python objects into and from BLOB.

“Attribute in AttributeList” Implementation

- Python style for-in loop.
- Iterator protocol implementation.
  - Iter() method for AttributeList class.
  - Iter() & next() for AttributeListIterator class.
- Allows following two forms of iterations:
- It = iter(AttList1)

```
For i in it:
  print i.data()
For attribute in AttList1:
  print attribute.data()
```

Inheritance implementation

- Single Inheritance
  - IQuery & IQueryDefinition
  - IBulkOperationWithQuery & IBulkOperation
  - IViewFactory & IQueryDefinition
- Multiple Inheritance
  - TableDescription, ITableDescription, ISchemaEditor classes

…Contd

- Whenever a class inheriting from a base class is “init” ed, all the classes (base classes + inheriting) are initialized.
- Whenever the inheriting class object goes out of scope its “dealloc” method, DECREFs all the base classes.
- The parent of the base classes have to be Py_NONE, to take care of the parent child implementation, which exists in all the classes.

Testing Techniques

- RefCounting
  - Requires Python recompilation with DEBUG option.
  - Carried out for all unit tests and integration test code.
- Valgrind memcheck
  - Only carried out for all the CoralBase classes and methods.
  - Not possible for RelationalAccess because of presence of SEAL code.

Present Status & Future

- The first release of PyCoral was made along with CORAL release 1.6.3.
- Subsequent releases of CORAL will all have PyCoral release also.

Conclusion

- ***Python/C API technique, provides an easy way to pythonize C++ classes, allowing both the python style and semantics, to be incorporated, without changing the underlying C++ code.***

Thank You all….