Python Web Development

By Dr. Tanima

Quick URLs:

- http://python.org/
- http://webware.sourceforge.net/
- http://www.python.org/cgi-bin/moinmoin/WebProgramming
- http://www.djangoproject.com
- https://pypi.python.org
- https://www.djangopackages.com

What is persistence?

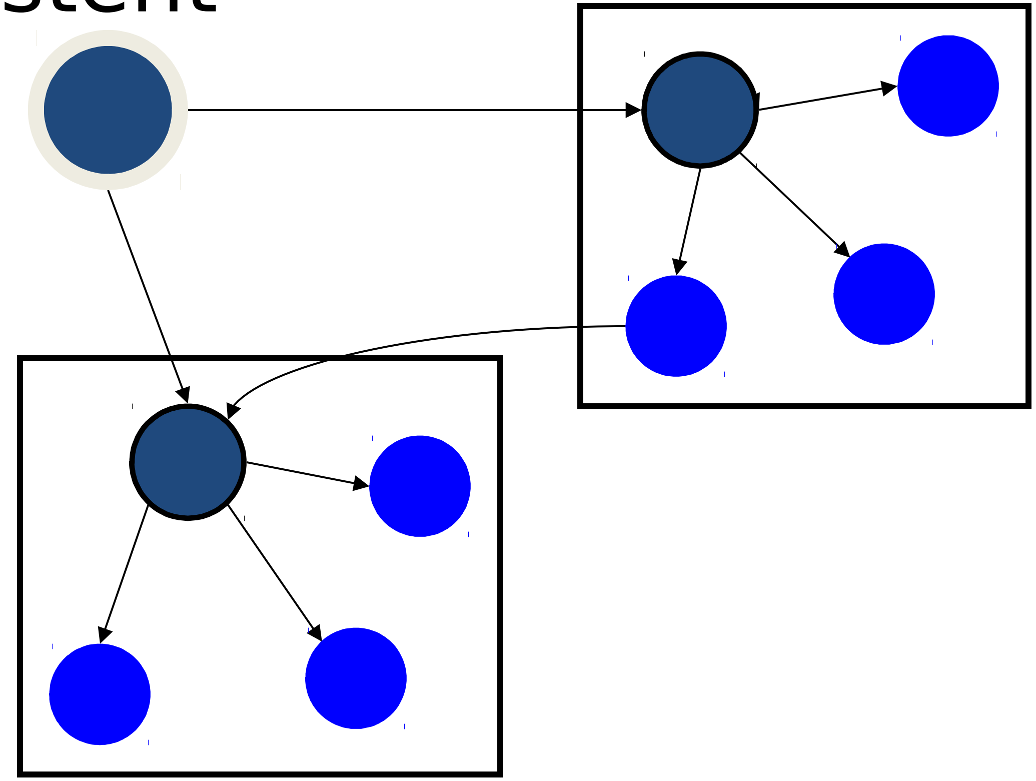
- Data lives longer than programs
  - Files, pipes, relational databases, etc.
  - But, discontinuity of representation : object serialization
- Orthogonal persistence
  - Automatic management of program state
  - Independent of data type or longevity
  - Allow programmer to focus on application data model

Python approach

- Goals
  - Minimize changes to existing programs
  - Work with standard interpreter
- Zope Object Database (ZODB)
  - Support Zope application server
  - Originally targeted for web development
  - Zope (Z Object Publishing Environment)
  - Any object reachable from ZODB root is persistent
  - Other App Server is Webware
- Separate database and storage layers
  - Store object's data in formatted text file, such as a CSV file.
  - Or can use a relational database, such as Gadfly, MySQL, PostgreSQL, or DB2.
  - These file formats and databases are well established, and Python has robust interfaces for all of these storage mechanisms.

Persistence by reachability

- Any object reachable from ZODB root is persistent



ZOPE: A simple example

```
from Persistence import Persistent
from Transaction import get_transaction
from ZODB.FileStorage import DB

class Counter(Persistent):
    _value = 0
    def inc(self):
        self._value += 1

def main():
    fs = DB(“data.fs”)
    conn = db.open(); root = conn.root()
    obj = root[“myobj”] = Counter()
    get_transaction().commit()
    obj.inc()
    get_transaction().commit()
```

Object serialization

- Standard pickle library
  - Serializes arbitrary object graph
  - Raises TypeError for sockets, files, etc.
  - Instance vars serialized via dictionary
- Hooks to define custom state
  - __getstate__() / __setstate__()
  - Persistent mixin ignores _v_ attributes

Pickling persistent objects

- Stores objects in separate records
  - Persistent objects pickled as oid + class
  - Works with cache to maintain identity
- Handling non-persistent objects
  - Copied into record of containing object
  - Sharing by persistent objects is problematic

Object identity / caching

- Cache maintains oid 🔔 object mapping
  - Guarantees only one copy of object
  - Unpickler loads all referenced objects
- Ghost objects
  - A webkit web client written in python
  - Only Persistent header initialized
  - No instance state loaded
  - State loaded on first object access
  - from ghost import Ghost
- LRU cache of recent objects

Attribute access handlers

- Persistent implements C wrappers
  - Override tp_getattro, tp_setattro slots
  - Mediate access to instance variables
  - Crucial Python feature

Transactions

- Supports multiple threads, processes
  - Independent database connections
  - Updates visible at transaction boundaries
- Optimistic concurrency control
  - When conflict occurs, abort and retry
- On error, abort to restore consistency
  - Reverts to last saved state

Concurrency and conflicts

- Invalidations sent at commit time
  - Clients process at transaction boundaries
- Conflicting transactions aborted
  - Write conflict at commit time
  - Read conflict on object access
- Application must retry on conflict
  - Can use generic wrapper
  - Can define conflict resolution method

Other features

- Undo support
  - Storage stores multiple revisions
  - Transactional undo reverts to earlier state
- BTrees: efficient persistent containers
- Storing code in database

Limitations

- Schema evolution
  - Must code manually in __setstate__()
- Database management
  - Manual pack() to remove revisions, do GC
- Sharing of non-persistent objects
- Integration with legacy code
  - Multiple inheritance helps
  - Factory classes