R"""Name server

Author : Anders Andersen
Created On : Thu Aug 27 09:25:47 1998
Last Modified By: Anders Andersen
Last Modified On: Fri Sep 10 13:34:49 1999
Status : Unknown, Use with caution!

The name server provides to basic types of services: export and import. The `exportIRef` method register the given interface reference with the given key in the name server. The `lookupIRef` method can then later be used to get access to this interface reference. The key used is usually a text string, but it can be anything and the interface reference could be a capsule reference (capsule proxy).

The name server is accessed through a name server proxy. It forwards the export and import operations to the actual name server. It also enhance the interface of the name server with implicit bindings. the `exportIRef` and `lookupIRef` methods is just a forwarding of the name server method calls. The new method `importIRef` creates an implicit binding to the actual interface. The result is that the returned interface reference can be used directly to access methods in the exported interface.

The name server proxy also provides the `exportCapsule` and `importCapsule` methods used to provide and gain access to a remote capsule.

If this name server module is run as a program, it will create a name server and start a serving loop where it is ready to accept request from name server proxies.

"""

```
# We need to check the type of some attributes
from types import *

# For low level communication
from socket import *

# Mainly to grab information about exceptions
import sys

# OS related functionality
import os

# Misc values for the Open-ORB core
from misc import *

# Message object for the Open-ORB core
from msg import *

# Local bindings (and interface references)
from lbind import *

# Remote bindings
#from opbind import *


class NameServerException(OpenORBException):
    R"""Name server exception

    All exceptions or errors introduced by the nameserver module is handled by this exception class.

    """
    pass
```

```
class NameServer:
    R"""The name server

    The name server can register interface references (and other references) with a given key (usually a text
    string, but it can be anything).

    """

    def __init__(self, port=nsPort):
        R"""Initialize and serve loop

        Initialize the name server and run the serving loop.

        """

        # Initialize data structures
        debug("Nameserver at %s:%d created" % (gethostname(), port))
        self.op = {"exportIRef": self.exportIRef,
                   "lookupIRef": self.lookupIRef,
                   "listIRefs": self.listIRefs,
                   "delIRef": self.delIRef,
                   "exportCapsule": self.exportCapsule,
                   "importCapsule": self.importCapsule,
                   "listCapsules": self.listCapsules,
                   "delCapsule": self.delCapsule}
        self.exportedIRefs = {}
        self.exportedCapsules = {}

        # Create and initialize listen object
        self.listen = Msg(gethostname(), port, 1)

        # The main (serving) loop
        while 1:

            # Recieve a request
            connection, requests = self.listen.recvreq()
            for req in requests:
                debug("Name server request: %s" % (‘req‘,))

                # Terminate?
                if req["op"] == "terminate":
                    return

                # Perform the request and send a reply (possible an error)
                try:
                    if not req.has_key("args"): req["args"] = ()
                    if not req.has_key("kw"): req["kw"] = {}
                    rep = apply(self.op[req["op"]], req["args"], req["kw"])
                except Exception:
                    (exc, val, tb) = sys.exc_info()
                    debug("Name server exception")
                    self.listen.sendrep(connection, ErrorObject(exc, val, tb))
                else:
                    self.listen.sendrep(connection, rep)

    def __del__(self):
        R"""Name server deleted

        Clean up before the name server is deleted.

        """
        debug("Name server at %s deleted" % (gethostname(),))
```

```
        del self.listen

    def _export(self, key, iref, exported):
        R"""Export an interface reference or capsule

        Export an interface reference or capsule with the given key.

        """
        if exported.has_key(key):
            str = "Name server export: key %s exists" % ('key',)
            debug(str)
            raise NameServerException, str
        else:
            exported[key] = iref
            debug("Name server export: %s exported" % ('key',))

    def exportIRef(self, key, iref):
        R"""Export an interface reference

        Export an interface reference with the given key.

        """
        self._export(key, iref, self.exportedIRefs)

    def exportCapsule(self, key, caps):
        R"""Export a capsule

        Export a capsule with the given key.

        """
        self._export(key, caps, self.exportedCapsules)

    def _del(self, key, exported):
        R"""Delete interface reference or capsule

        Delete the exported interface reference or capsule with the given key.

        """
        if exported.has_key(key):
            del exported[key]
        else:
            str = "Name server del: %s doesn't exists" % ('key',)
            debug(str)
            raise NameServerException, str

    def delIRef(self, key):
        R"""Delete interface reference

        Delete the exported interface reference with the given key.

        """
        self._del(key, self.exportedIRefs)

    def delCapsule(self, key):
        R"""Delete capsule

        Delete the exported capsule with the given key.

        """
        self._del(key, self.exportedCapsules)

    def _lookup(self, key, exported):
        R"""Look up an interface reference or capsule

        Look up an interface reference with the given key.

        """
        if exported.has_key(key):
```

```
            return exported[key]
        else:
            str = "Name server lookup: %s doesn't exists" % ('key',)
            debug(str)
            raise NameServerException, str

    def lookupIRef(self, key):
        R"""Look up an interface reference

        Look up an interface reference with the given key.

        """
        return self._lookup(key, self.exportedIRefs)

    def importCapsule(self, key):
        R"""Look up a capsule

        Look up a capsule with the given key.

        """
        return self._lookup(key, self.exportedCapsules)

    def listIRefs(self):
        R"""List all exported interface references

        List all exported interface references in this nameserver.

        """
        return self.exportedIRefs

    def listCapsules(self):
        R"""List all exported capsules

        List all exported capsules in this nameserver.

        """
        return self.exportedCapsules


class NameServerProxy:
    R"""A name server proxy

    An instance of this class is used to access a name server. The name server is identified by a node (host)
    and a communication port. A name server proxy can also create implicit bindings to interfaces.

    """

    def __init__(self, node="", port=nsPort):
        R"""Initialize a name server proxy

        The name server proxy is initialized with a node (host) and a communication port.

        """
        if node:
            self.message = Msg(node, port)
        else:
            self.message = Msg(gethostname(), port)

    def stopserve(self):
        R"""Terminate the name server

        Terminate the name server associated with this name server proxy.

        """
        self.message.announce({"op": "terminate"})

    def exportIRef(self, key, iref):
```

```
    R"""Export interface reference                                          277

    Export an interface reference with the given key to the name server.

    """
    self.message.message({"op": "exportIRef", "args": (key, iref)})         283
                                                                            284
def exportCapsule(self, key, caps):                                         285
    R"""Export capsule                                                      286

    Export capsule with the given key.

    """
    self.message.message({"op": "exportCapsule", "args": (key, caps)})      291
                                                                            292
def delIRef(self, key):                                                     293
    R"""Delete exported interface reference                                 294

    Delete interface reference with the given key from the nameserver.

    """
    self.message.message({"op": "delIRef", "args": (key,)})                 300
                                                                            301
def delCapsule(self, key):                                                  302
    R"""Delete exported capsule                                             303

    Delete capsule with the given key from the nameserver.

    """
    self.message.message({"op": "delCapsule", "args": (key,)})              308
                                                                            309
def lookupIRef(self, key):                                                  310
    R"""Look up an interface reference                                      311

    Look up the interface reference with the given key at the name server.

    """
    return self.message.message({"op": "lookupIRef", "args": (key,)})       317
                                                                            318
def importIRef(self, key):                                                  319
    R"""Import an interface reference                                       320

    Import the interface reference with the given key and create an implicit binding to it.

    """
    from opbind import *                                                    326
    siref = self.message.message({"op": "lookupIRef", "args": (key,)})      327
    ciref = IRef(None, siref.__impID__, siref.__expID__)                    328
    remoteBind(ciref, siref)        # Ignore returned binding object        329
    return ciref                                                            330
                                                                            331
def importCapsule(self, key):                                               332
    R"""Import a capsule reference                                          333

    Returns a capsule reference (a capsule proxy) to the capsule with the given key.

    """
    return self.message.message({"op": "importCapsule", "args": (key,)})   339
                                                                            340
def listIRefs(self):                                                        341
    R"""List interface references                                           342

    List all exported interface references from the nameserver.

    """
    return self.message.message({"op": "listIRefs"})                        347
                                                                            348
def listCapsules(self):                                                     349
```

```
        R"""List capsules                                                    350

        List all exported capsules from the nameserver.

        """
        return self.message.message({"op": "listCapsules"})                 355
                                                                             356
    def capsuleFromIRef(self, iref):                                         357
        R"""Get capsule from interface reference                             358

        Get the capsule (proxy) from an interface reference. If iref is not an instance of the IRef class
        then iref is used as the key for a lookup after the actual interfce reference from the name server.

        """
        if not isinstance(iref, IRef):                                       367
            iref = self.lookupIRef(iref)                                     368
        return iref.__local__["object"]["capsule"]                           369
                                                                             370
                                                                             371
# Create the name server if this is run as a program                         372
if __name__ == "__main__":                                                   373
    if len(sys.argv) > 1:                                                    374
        import string                                                        375
        ns = NameServer(string.atoi(sys.argv[1]))                            376
    else:                                                                    377
        ns = NameServer()                                                    378
                                                                             379
                                                                             380
# LocalWords: Aug Oct UK NORUT misc lbind rbind NameServerException def init 381
# LocalWords: OpenORBException aacodefont nameserver NameServer nsPort op AF 382
# LocalWords: gethostname exportIRef lookupIRef listenSocket INET addr req kw 383
# LocalWords: gethostbyname unmarshall recv BUFSIZ args exc msg tb marshall  384
# LocalWords: ErrorObject del iref NameServerProxy proxySocket IRef          385
# LocalWords: isinstance exportCapsule CapsuleProxy importIRef siref ciref ns 386
# LocalWords: impID expID remoteBinding importCapsule len argv atoi          387
```