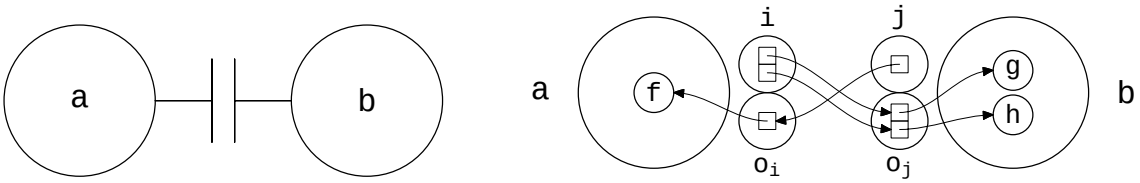
R"""Local bindings

Author : Anders Andersen
Created On : Wed Mar 18 22:18:34 1998
Last Modified By: Anders Andersen
Last Modified On: Fri Apr 7 17:09:51 2000
Status : Unknown, Use with caution!

This module implements a local binding with the `IRef` and the `IObj` classes. An interface reference (created from the `IRef` class) has one export and one import interface description. The interface reference is bound to a local object and the export interface must match this local object (the methods must exist and match). You make a local binding between two interfaces with the `localBind` function. You break (disconnect) a local binding with the `break` method of the local binding (control) object or the `breakLB` library function.



The figure above gives an example of a local binding. We often draw local bindings between two objects as shown on the left part of the figure. The right part shows the details of this implementation of local bindings. The objects `a` and `b` have one interface each (shown as the interface references `i` and `j` and the interface objects o_i and o_j). The interface objects (created from the `IObj` class) forward method calls to the local object. In this example, `a` exports `f` and imports `g` and `h`, and `b` exports `g` and `h` and imports `f`:

```
i = IRef(a, ["f"], ["g", "h"])
j = IRef(b, ["g", "h"], ["f"])
```

We create a binding between the methods imported and exported in interface `i` and `j` with a `localBind` call:

```
lb = localBind(i, j)
```

`a` can now access methods `g` and `h` from `b` with the interface reference `i` and `b` can access method `f` from `a` with the interface reference `j`. `localBind` returns a control object for the local binding. This is an example of `a` calling the method `g` in `b` (with the argument `"hello"`):

```
i.g("hello")
```

Be aware that a local binding will not automatically break if its local binding (control) object is deleted or garbage collected. The local binding between interfaces `i` and `j` will still exist if their control object `lb` is deleted (garbage collected). The local binding between interface `i` and `j` is broken (disconnected) with one of these two calls:

```
lb.breakBinding()
breakBinding(i, j)
```

"""

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

```
# Miscellaneous definitions and values common for the Open-ORB core
from misc import *


class LBindException(OpenORBException):
    R"""Local bind exception

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

    """
    pass


class IMethod:
    R"""Forwarding interface methods

    An instance of this class is callable and it forwards a method call to the given bounded method.

    """

    def __init__(self, object=None, key=""):
        R"""Save method reference

        We need to save a reference to the method so we can call it later.

        """
        self.object = object
        self.key = key

    def __call__(self, *args, **kw):
        R"""Call the bounded method

        This method is called when an instance of this class is used as a method. We just forwards the call
        to the actual method supplied with the the given arguments. The result is returned to the caller.

        """
        return apply(getattr(self.object, self.key), args, kw)


class IObj:
    R"""An interface object

    Used to forward method calls to the actual object. Forwarding methods (see the IMethod class) are
    inserted in the name space of instances of this class when an IRef object is initialized.

    """
    pass


class IRef:
    R"""An interface reference

    Local bindings are made between interfaces. An interface reference contains an interface description for
    the exported and the imported methods (__expID__ and __impID__) and references to the actual objects
    (__local__ and __remote__). The exported interface description is tested against the implementation of
    the local object when the interface is created. The imported interface description is tested against the
    imported interface when a local bind is done. After a local bind is done, the interface reference contains
    forwardings of the exported functions of the remote object. Dummy interface references can be used as
    proxy objects for objects with an interface with opposite export/import interface description. An example
    of this is given in the documentation of the composite module (the interface reference i).

    """

    def __init__(self, object=None, expID=[], impID=[]):
```

R"""Initialize the interface reference

Test the exported interface description against the implementation and save some information about the interface and the local object. The `object` attribute is either the actual object (for local irefs) or a dictionary containing `"capsule"` (a capsule or a capsule proxy), `"comp"` (the name of the registered component) and `"iface"` (the name of the interface of the registered component). The `object` attribute can also have the value `None`, which means that it is used as a proxy (`expID` should then be empty).

```
    """
    self.__local__ = {}
    self.__local__["object"] = object
    if object:
        if not type(object) is DictType:
            self.__testExpInterface__(expID)
    self.__expID__ = expID
    self.__remote__ = {}
    self.__impID__ = impID

def __fetchIDKeys__(self, ID):
    R"""Fetch the keys from an interface description

    This returns the keys (method names) from an interface description.

    """
    return ID       # Currently ID contains only the keys of the methods

def __expIDKeys__(self):
    R"""Fetch the keys from exported interfaces

    This returns the keys (method names) from the interface description of the exported interfaces.

    """
    return self.__fetchIDKeys__(self.__expID__)

def __impIDKeys__(self):
    R"""Fetch the keys from imported interfaces

    This returns the keys (method names) from the interface description of the imported interfaces.

    """
    return self.__fetchIDKeys__(self.__impID__)

def __testExpInterface__(self, expID):
    R"""Test exported interface description

    Does the object satisfies the given export interface? Give an error exception if it doesn't. All methods are forwarded through the interface object.

    """
    self.__local__["iobj"] = IObj()
    for key in self.__fetchIDKeys__(expID):
        if not hasattr(self.__local__["object"], key):
            raise LBindException, \
                  "IRef error: exported method doesn't exists: %s" \
                  % (key,)
        if not callable(getattr(self.__local__["object"], key)):
            raise LBindException, \
                  "IRef error: exported object not callable: %s" % (key,)
        self.__local__["iobj"].__dict__[key] = IMethod(
            self.__local__["object"], key)

def __testImpInterface__(self, iref):
```

```
        R"""Test imported interface description                                    237

        Does the exported interface satisfies the imported interface description of the given (remote) interface
        reference (iref)? Inserts forwardings in the remote interface reference to the methods in the interface
        object if it does. Give an exception otherwise. Also save information about the local object in the
        remote interface reference.

        """
        for key in iref.__impIDKeys__():                                           247
            if not hasattr(self.__local__["iobj"], key):                           248
                raise LBindException, \                                            249
                      "IRef error: export/import ID mismatch: %s" % (key,)         250
            iref.__dict__[key] = IMethod(self.__local__["iobj"], key)              251
        iref.__remote__["object"] = self.__local__["object"]                       252
                                                                                   253
    def __breakBinding__(self):                                                    254
        R"""Break the binding to other interface                                   255

        Break the binding to another interface. This must be done on both interfaces (see breakBinding
        below).

        """
        for key in self.__impIDKeys__():                                           261
            del self.__dict__[key]                                                 262
        del self.__remote__["object"]                                              263
                                                                                   264
                                                                                   265
class CIRef(IRef):                                                                 266
    R"""Client interface reference                                                 267

    An interface reference only importing methods (for clients).

    """
    def __init__(self, object=None, impID=[]):                                     272
        IRef.__init__(self, object, [], impID)                                     273
                                                                                   274
                                                                                   275
class SIRef(IRef):                                                                 276
    R"""Server interface reference                                                 277

    An interface reference only exporting methods (for servers).

    """
    def __init__(self, object=None, expID=[]):                                     282
        IRef.__init__(self, object, expID, [])                                     283
                                                                                   284
                                                                                   285
class SrcRef(IRef):                                                                286
    def __init__(self, object=None):                                               287
        IRef.__init__(self, object, ["put"], [])                                   288
                                                                                   289
                                                                                   290
class SinkRef(IRef):                                                               291
    def __init__(self, object=None):                                               292
        IRef.__init__(self, object, [], ["put"])                                   293
                                                                                   294
                                                                                   295
class SigSrcIRef(IRef):                                                            296
    R"""Signal source interface reference                                          297

    Interface reference for the source of a signal. A call to the event method of a no-bound signal source
    inteface will be ignored.

    """
                                                                                   304
```

```
    def __init__(self, object=None):
        R"""Initalize the signal source interface reference

        Initalizing the signal source interface reference through the IRef class with signal source specific
        values.

        """
        IRef.__init__(self, object, [], ["event"])

    def __getattr__(self, key):
        R"""Ignoring event calls

        Ignoring calls to a non-exsistig event method. This will be called when a signal is sent to a non-
        bound interface.

        """
        if key == "event":
            return _emptyMethod

class SigSinkIRef(IRef):
    R"""Signal sink interface reference

    Interface reference for the sink of a signal.

    """
    def __init__(self, object=None):
        R"""

        Initalizing the signal sink interface reference through the IRef class with signal sink specific values.

        """
        IRef.__init__(self, object, ["event"], [])


class LBindCtrl:
    R"""Local binding control

    Control object for a local binding.

    """

    def __init__(self, capsule=None, iref1=None, iref2=None):
        R"""Initialize the local binding control

        Save some informations about the local binding.

        """
        self.capsule = capsule
        self.iref1 = iref1
        self.iref2 = iref2

    def breakBinding(self):
        R"""Break this local binding

        Break this local binding without removing this object (the local binding control object).

        """
        if self.capsule:
            self.capsule.breakBinding(self.iref1, self.iref2)
        else:
            self.iref1.__breakBinding__()
            self.iref2.__breakBinding__()
        self.iref1 = None
        self.iref2 = None

    def reBind(self, iref1=None, iref2=None):
```

```
        R"""Reestablish a local binding                                    374

        Reestablish a local binding between two (new) interfaces.

        """
        if self.capsule:                                                   379
            self.capsule.localBind(iref1, iref2)                           380
        else:                                                              381
            iref1.__testImpInterface__(iref2)                              382
            iref2.__testImpInterface__(iref1)                              383
        self.iref1 = iref1                                                 384
        self.iref2 = iref2                                                 385
                                                                           386
    def reBindOneWay(self, iref1=None, iref2=None):                        387
        R"""Reestablish a one-way local binding                            388

        Reestablish a one-way local binding between two (new) interfaces.

        """
        if self.capsule:                                                   393
            self.capsule.localBindOneWay(iref1, iref2)                     394
        else:                                                              395
            iref2.__testImpInterface__(iref1)                              396
        self.iref1 = iref1                                                 397
        self.iref2 = iref2                                                 398
                                                                           399
                                                                           400
def localBind(iref1, iref2):                                               401
    R"""Create a local binding                                             402

    Create a local binding between the two interfaces iref1 and iref2. This includes a compatibility check
    of the interfaces. localBind returns a control object for the binding.

    """
    if (not type(iref1.__local__["object"]) is DictType and                410
        not type(iref2.__local__["object"]) is DictType):                  411
        iref1.__testImpInterface__(iref2)                                  412
        iref2.__testImpInterface__(iref1)                                  413
        return LBindCtrl(None, iref1, iref2)                               414
    elif (type(iref1.__local__["object"]) is DictType and                  415
          type(iref2.__local__["object"]) is DictType):                    416
        return iref1.__local__["object"]["capsule"].localBind(iref1, iref2)  417
    else:                                                                  418
        raise LBindException, \                                            419
              "localBind: can not bind local and non-local irefs"          420
                                                                           421
                                                                           422
def localBindOneWay(iref1, iref2):                                         423
    R"""Create a one-way local binding                                     424

    Create a one-way local binding from interface iref1 to interface iref1, meaning that you can use
    interface reference iref1 to call methods exported through interface reference iref2. localBind
    returns a control object for the binding.

    """
    if (not type(iref1.__local__["object"]) is DictType and                433
        not type(iref2.__local__["object"]) is DictType):                  434
        iref2.__testImpInterface__(iref1)                                  435
        return LBindCtrl(None, iref1, iref2)                               436
    elif (type(iref1.__local__["object"]) is DictType and                  437
          type(iref2.__local__["object"]) is DictType):                    438
        return iref1.__local__["capsule"].localBindOneWay(iref1, iref2)    439
    else:                                                                  440
        raise LBindException, \                                            441
```

```
                "localBind: can not bind local and non-local irefs"
```

```
def breakBinding(iref1, iref2):
    R"""Break a local binding

    Break the local binding between iref1 and iref2.

    """
    if (not type(iref1.__local__["object"]) is DictType and
        not type(iref2.__local__["object"]) is DictType):
        if ((iref1.__local__["object"] == iref2.__remote__["object"]) or
            (iref2.__local__["object"] == iref1.__remote__["object"])):
            iref1.__breakBinding__()
            iref2.__breakBinding__()
        else:
            raise LBindException, \
                  "breakBinding: can not break binding between interfaces " + \
                  "that are not bound"
    elif (type(iref1.__local__["object"]) is DictType and
          type(iref2.__local__["object"]) is DictType):
        iref1.__local__["object"]["capsule"].breakBinding(iref1, iref2)
    else:
        raise LBindException, \
              "breakBinding: can not break binding between local and " + \
              "non-local irefs"
```

```
#   LocalWords:  aacodefont localBind LBindException lbind expID UK Jul args Oct
#   LocalWords:  forwardings impID def init testExpInterface getattr IObj IRef
#   LocalWords:  testImpInterface hasattr dict hfil Apr LocalWords NORUT mbox
#   LocalWords:  iref iobj attr texttt riptsize IMethod localBindOneWay Aug misc
#   LocalWords:  irefs kw includegraphics OpenORBException comp iface DictType
#   LocalWords:  fetchIDKeys expIDKeys impIDKeys LBindCtrl elif
```