

OrbixWeb

- Corba implementation
 - From IONA
 - Using Java
 - Variant of ORBIX
- Today
 - Programming example (from OrbixWeb v2)
 - Features of OrbixWeb
 - Compared with Corba2 standard

Grid example

- Steps
 - Define the interfaces, using IDL
 - Implement interfaces with Java classes
 - Write a server application which creates instances of the classes and then informs OrbixWeb when initialisation has been done and the server is ready to accept requests
 - Register the server
 - Write a client application to connect to the server and to use the server's objects
- Grid (matrix) of int values
 - IDL:

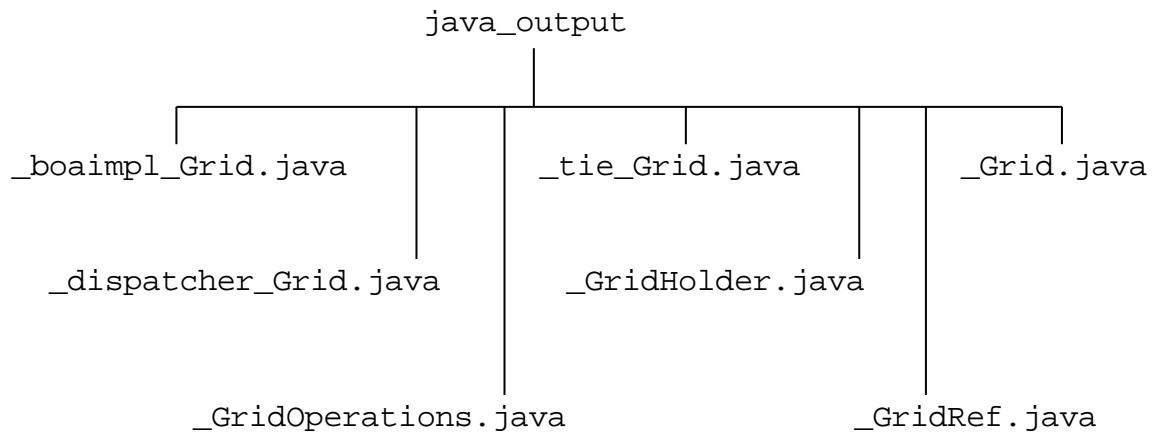
```
// In file grid.idl.  
interface Grid {  
    readonly attribute short height;  
    readonly attribute short width;  
  
    void set(in short n, in short m, in long value);  
    long get(in short n, in short m);  
};
```
 - Get/set an element of grid

From IDL to java

- Run idl compiler

```
idl grid.idl
```

- Generates lots of files



<code>_GridRef</code>	Java client view of the IDL interface (interface)
<code>Grid</code>	Implementation for <code>_GridRef</code> , stub/proxy
<code>_GridHolder</code>	Container for a <code>Grid</code> , used with in/out parameters
<code>_GridOperations</code>	Mapping of attributes/operations to Java methods (interface). Server provides implementation class
<code>_boaimpl_Grid</code>	Server side abstract class, used to implement client interface, using BOA
<code>_tie_Grid</code>	idem, but using different (TIE) approach
<code>_dispatcher_Grid</code>	Server side skeleton

Grid reference

- Client view: `_gridRef.java`

```
public interface _gridRef extends
    IE.Iona.Orbix2.CORBA._ObjectRef {
    public short get_height()
        throws IE.Iona.Orbix2.CORBA.SystemException ;

    public short get_width()
        throws IE.Iona.Orbix2.CORBA.SystemException ;

    public void set(short n, short m, int value)
        throws IE.Iona.Orbix2.CORBA.SystemException ;

    public int get(short n, short m)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
}
```

- Stub: `grid.java`

```
public class grid
    extends IE.Iona.Orbix2.CORBA.BaseObject
    implements _gridRef
{
    ...
    public grid (String _objref)
        throws IE.Iona.Orbix2.CORBA.SystemException
    ...
    public short get_height()
        throws IE.Iona.Orbix2.CORBA.SystemException
    ...
    public void set(short n, short m, int value)
        throws IE.Iona.Orbix2.CORBA.SystemException
    ...
}
```

Stub

- A little bit of the internals

```
public int get(short n, short m)
    throws IE.Iona.Orbix2.CORBA.SystemException
{
    int IT_result = 0;
    short locateAttempts = _CORBA.IT_LOCATE_ATTEMPTS;
    boolean relocate;
    do {
        relocate = false;
        IE.Iona.Orbix2.CORBA.Request _mb = new
            IE.Iona.Orbix2.CORBA.Request(this,
                "get", false);
        _mb.insertShort(n);
        _mb.insertShort(m);
        try {
            _mb.invoke();
            IT_result = _mb.extractLong();
            _mb.extractOutParams();
            _mb.inReplyPostMarshal();
        } catch (IE.Iona.Orbix2.CORBA.IIOP _ex) {
            if (_ex.minor() == 10601) {
                IE.Iona.Orbix2.CORBA.IOR _ior = _ex.getIOR();
                _setIOR(_ior);
                relocate = true;
            } else {
                throw _ex;
            }
        } catch (IE.Iona.Orbix2.CORBA.SystemException _ex) {
            throw _ex;
        } catch (IE.Iona.Orbix2.CORBA.CORBAException _ex) {
            throw new IE.Iona.Orbix2.CORBA.UNKNOWN(12001,
                IE.Iona.Orbix2.CORBA.CompletionStatus.MAYBE);
        }
    } while (relocate && --locateAttempts > 0);
    return IT_result;
}
```

Server side

- Functionality interface: `_gridOperations.java`

```
public interface _gridOperations {
    public short get_height()
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    ...
    public void set(short n, short m, int value)
        throws IE.Iona.Orbix2.CORBA.SystemException ;
    ...
}
```

- Almost similar to interface (`_gridRef`) for client

Server implementation

- GridImplementation

```
class GridImplementation implements _gridOperations {

    public int m_height;    // store the height
    public int m_width;    // store the width
    public long m_a[][];   // a 2D array to hold the grid
                          data

    public GridImplementation(int height, int width)
        throws SystemException
    {
        m_a = new long[height][width]; // allocate the 2D
            array
        m_height = (short)height; // set up height
        m_width = (short)width;   // set up width
    }

    public short get_height() {
        return (short)m_height;
    }
    ...
    public void set(short n, short m, int value) {
        m_a[n][m] = value;
    }
    ...
}
```

- Provides object creation & functionality

Server creation

- Letting the world know about the object

```
public class javaserver1 {  
  
    public static void main(String args[]) {  
  
        _gridRef gridImpl = null;  
  
        try {  
            gridImpl = new _tie_grid (new  
                GridImplementation(100,100));  
        }  
        catch(SystemException se) {  
            ...  
            return;  
        }  
  
        try {  
            _CORBA.Orbix.impl_is_ready("GridSrv");  
        }  
        catch(SystemException se) {  
            ...  
            System.exit(1);  
        }  
    }  
}
```

TIE

- GridImplementation ‘only’ knows grid functionality
 - But no CORBA object functionality
- GridImplementation has to be wrapped into TIE object

```
public class _tie_grid extends grid {
    private _gridOperations m_impl;
    ...
    public short get_height()
        throws IE.Iona.Orbix2.CORBA.SystemException {
        return m_impl.get_height();
    }
    ...
}
```

- TIE object is a grid object
- But delegates to implementation

```
public _tie_grid(_gridOperations impl)
    throws IE.Iona.Orbix2.CORBA.SystemException {
    super();
    this.m_impl = impl;
    _objMgr(new
        IE.Iona.Orbix2.CORBA.ServerObjectMgr(_di
            spatcher, "grid", null, null, this));
}
```

- And has told environment about itself

Compiling & installing

- Making it work
 - Environment specific
 - Here: JDK1.0.2 on Windows95, OrbixWeb2.0
- Compile
- Start Orbix daemon (name server)
`start ...\orbixd`
- Register server
`putit -j GridSrv gridtest.javaserver1`
 - Java, name of server, specific class
- Using the object from a client ...

Client example

- Ask grid class for binding and use...

```
public class javaclient1 {

    public static _gridRef p = null;

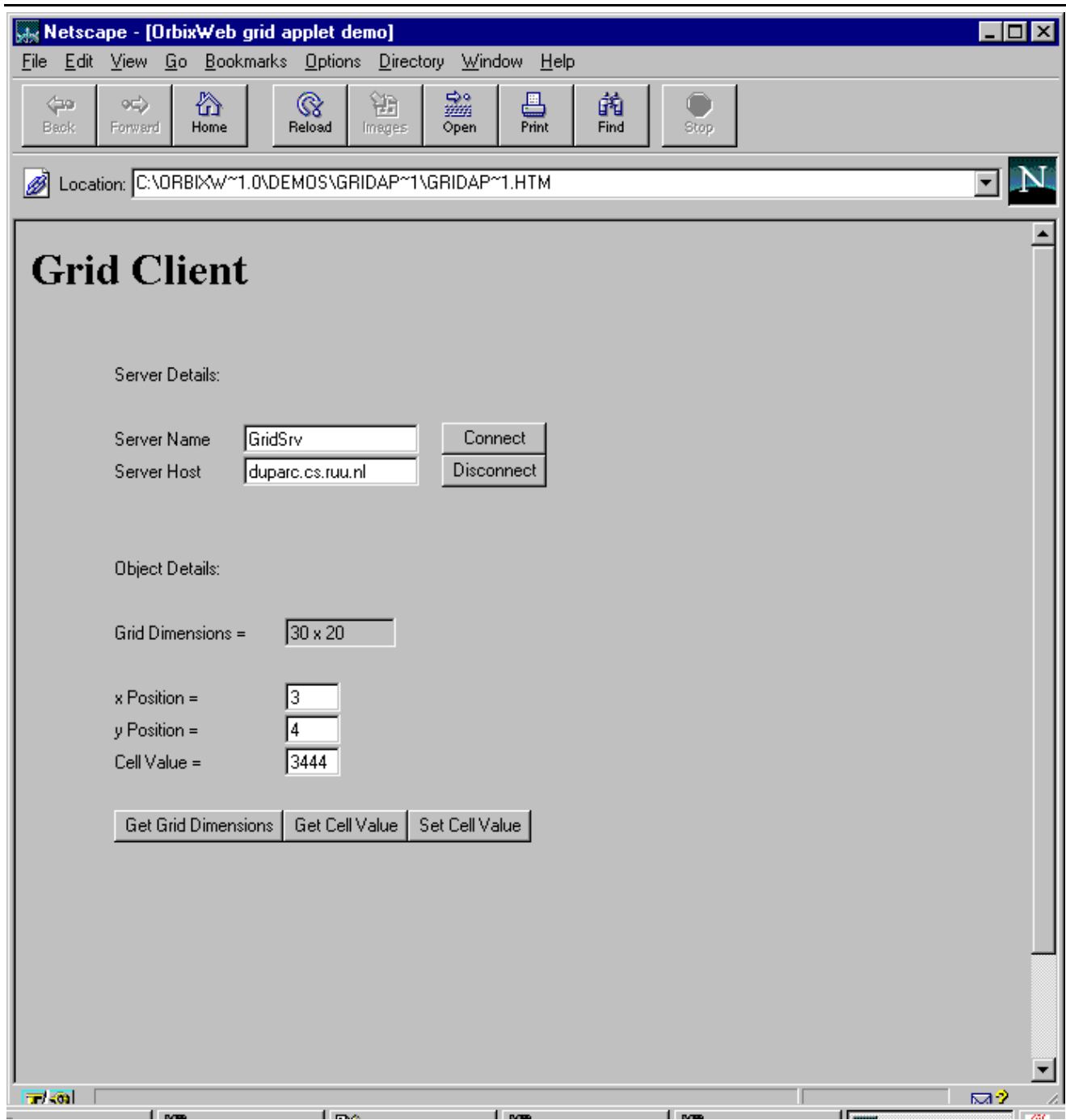
    public static void main(String args[]) {
        String hostname = ... ;
        ...
        try {
            p = grid._bind(":GridSrv", hostname);
        }
        catch (SystemException ex) {
            ...
        }

        do_width_height(p);
        do_set_get(p);
        do_loop(p);
    }

    public final static void do_set_get(_gridRef p) {
        int val = 0;
        try {
            p.set((short)2, (short)4, 123);
            p.set((short)0, (short)0, 0);
            val = p.get((short)2, (short)4);
        } catch (SystemException ex) {
            ...
        }
    }
}
...
```

Client applet

- From examples



IDL

- Java/C++ alike

- Describing interfaces

```
interface Account {  
    attribute float balance;  
    readonly attribute string owner;  
  
    void makeDeposit  
        (in float amount, out float newBalance);  
    void makeWithdrawal  
        (in float amount, out float newBalance);  
};
```

- Attributes

- Interface methods, directional parameters

- Oneway

```
oneway void notice(in string notice);
```

- No return value

- Context

```
void op(in unsigned long s)  
    context("accuracy", "base");
```

- Extra parameter: String->String mapping

- Others: modules, exceptions, inheritance

- Types: basic, struct, array, any, ...

- Mapping to Java straightforward

Object identification

- Object reference
 - Object marker, identification within server
 - Server name, or implementation name
 - Server host
 - IDL interface type of the object.
 - Interface Repository (IR) server in which the definition of this interface is stored.
 - IR server host.
- All is specified implicitly (e.g. by location) or explicitly (parameter)
- Example

```
p = grid._bind(":GridSrv", hostname);
System.out.println( "OBJ STR " +
                    p._object_to_string() ) ;
```

 - Gives

```
:\duparc.cs.ruu.nl:GridSrv:0::IR:grid
```
- Only usable by OrbixWeb implementations
 - Otherwise use IOR (Interoperable Object Reference)

Where to get references

- Via daemon, using `_bind()` at client side
- Via CORBA naming service
 - Hierarchical naming
- Via a string representation of an IOR, converting it back to an IOR
- Via parameter passing, returning values
 - Proxies are created automatically

Binding

- In a client with variants of `_bind()`
 - Accepts marker + server name with optional host name
 - Or complete IOR string
 - Interface is implicit since `_bind()` is invoked on interface: `grid._bind()`
- Idea:
 - Get certain (via marker) object in server on some host
 - Leaving out information: fall back on default behavior
- If no host is specified
 - A Locator is used

Locator

- Locator locates the 'right' object
- Default locator
 - Contacts local OrbixWeb daemon
 - Ask for a list of hosts for a server name
 - Try to find a valid server registration, return first one found
 - Otherwise try locally
- Default locator can be replaced
- Locator uses some configuration data
 - Fixed at compile time (class Configure)
 - Or found in configuration file

Configuration

- At least some information has to be known...

- E.g. invoking `orbixd -v` gives

```
Orbix daemon v2.1.0
```

```
s1230-2.1.0: Orbix Version v2.1.0 for JDK 1.0.2 on  
Win32
```

```
Implementation Repository Path
```

```
:C:\OrbixWeb2.0\config\Repository
```

```
Daemon Port :1570
```

```
Daemon IIOP Port :1571
```

```
Daemon Port Base :1800
```

```
Daemon Port Range :50
```

```
Orbix Errors File :C:\OrbixWeb2.0\config\ErrorMsgs
```

```
Orbix Locator Path :C:\OrbixWeb2.0\config
```

```
Interface Repository Path
```

```
:C:\OrbixWeb2.0\config\Interfaces
```

```
Local Host :duparc.cs.ruu.nl
```

```
Local domain :cs.ruu.nl
```

```
OrbixWeb 2.0 enabled.
```

```
Java Interpreter :C:\jdk102\java\bin\java.exe
```

```
Default Classpath
```

```
:C:\jdk102\java\lib\classes.zip;C  
:\OrbixWeb2.0\classes
```

Server activation mode

- Shared Activation Mode
 - Objects with same server name managed by same process
 - Activated on first call or as persistent server before any method call
 - Using implementation repository for mapping between server name and executables (i.e. class name + class path)
- Unshared Activation Mode
 - Individual objects are registered, each handled by one server process
- Per-method Call Activation Mode
 - Individual method names are registered, each invocation handles by a different server process

Server activation mode

- Secondary modes
- Per-client
 - New server for different client
- Per-client process
 - New server for different client process
- Multiple client
 - Shared between clients and client processes

Implementation repository

- Maintains relevant information about servers and executables, activation mode, ...
- Utilities to manipulate IR
 - putit: install server
- Others
 - catit Outputs full information about a given Implementation Repository entry
 - chmodit Allows launch and invoke rights on a server to be granted to users other than the server owner
 - chownit Allows the ownership of Implementation Repository entries and directories to be changed
 - killit Kills a running server process
 - lsit Lists a specific entry or all entries
 - mkdirit Creates a new registration directory
 - pingit Pings the OrbixWeb daemon to determine whether it is alive
 - psit Outputs a list of server processes known to the OrbixWeb daemon
 - rmdirit Removes a registration directory
 - rmit Removes an Implementation Repository entry or modifies an entry
- Other issues
 - Ownership, security, unregistered servers

ORB Interoperability

- General Inter-ORB Protocol (GIOP)
 - Data representation and message formats
 - Assuming connection-oriented communication
- Internet Inter-ORB Protocol (IIOP)
 - Variant for internet, using TCP/IP as transport layer
- GIOP data coding
 - Marshalling for datatypes (only basic ones)
- GIOP message types
 - For normal invocation: request, reply
 - Others: canceling, error, ...

Callback

- Client invokes server method
 - Server invokes client method back
 - Client normally waits for reply from server.
Deadlock?
- Consequence of client/server architecture
 - Blocking of client
- Solutions
 - Oneway server invocation does not block
 - Separate thread in client which processes events

```
public class EventProcessor extends Thread {
    public void run () {
        try {
            _CORBA.Orbix.processEvents
                (_CORBA.IT_INFINITE_TIMEOUT);
        }
        catch (SystemException se) {
            ...
        }
    }
}
```

Dynamic invocation

- Dynamically constructing requests
- Normally done statically
 - Via compiler, stubs
- Example: Account and Bank

```
interface Account {
    readonly attribute float balance;

    void makeDeposit(in float f);
    void makeWithdrawal(in float f);
};

interface Bank {
    exception Reject { string reason; };

    Account newAccount(in string owner,
                      inout float initialBalance)
        raises (Reject);
    void deleteAccount(in Account a);
};
```

- Static invocation

```
_BankRef bRef = ...
_AccountRef aRef;
aRef = bRef.newAccount("Chris", (float)1000.00);
```

Dynamic invocation

- Getting an object reference, constructing a request, invoking the request

```
_ObjectRef oRef =
    _CORBA.Orbix.string_to_object(refStr);

Request request = oRef._request("newAccount");

(request.arguments()
    .add(new Flags (_CORBA.ARG_IN))
    .value()
).insertString("Chris");

(request.arguments()
    .add(new Flags (_CORBA.ARG_INOUT))
    .value()
).insertFloat((float) 1000.00);

request.invoke ();
```

- Or using another programmatic interface for making requests

Typecodes

- All values and objects have explicit type code
- Usage example

- IDL

```
struct Example {  
    long l;  
};  
  
interface Test {  
    void op(in any a);  
};
```

- Passing 'any' type

- Passed as parameter

```
_TestRef tRef;  
Any a;  
tRef = Test._bind ();  
tRef.op (a);
```

- Runtime checking somewhere

```
IE.Iona.Orbix2.CORBA.TypeCode t (a.type());  
if(t.equals(Test._tc_Example)) {  
    ...  
}
```

Interface repository

- Interface Repository (IR)
 - Maintains information on IDL definitions
 - Separate server providing meta information
 - Interface defined via IDL itself
 - Maintains tree-like structure for definitions
- Usage example
 - Giving list of operation & attribute names

```
IE.Iona.Orbix2._InterfaceDefRef interfaceRef;  
  
// Get interface definition:  
interfaceRef = objRef._get_interface();  
  
IE.Iona.Orbix2.InterfaceDef.FullInterfaceDescription  
    full;  
  
full = interfaceRef.describe_interface();  
  
System.out.println ("The operation names are:");  
  
long i;  
  
for (i=0; i < full.operation.length; i++) {  
    System.out.println ("\t" +  
        full.operation.buffer[i].name);  
  
System.out.println ("The attribute names are:");  
  
for (i=0; i < full.attributes.length; i++) {  
    System.out.println ("\t" +  
        full.attributes.buffer[i].name);  
}
```