Introduction to CORBA

- Chia-Pin Huang

Overview

- Basic CORBA
- Basic IDL
- Examples
- Summary
- References
• **CORBA**
  – Common Object Request Broker Architecture

• **Why CORBA?**
  – Industrial Standard
    • In 1989, Object Management Group (OMG) tried to create a standard architecture (CORBA) for networked distributed objects.
    • A powerful language-neutral and platform-neutral technology for distributed objects.
  – Supports multiple implementation languages
    • Can help the communication between C++ and JAVA

• **History of the CORBA Programming Models in JAVA**
  – In 1997, Remote Method Invocation, or RMI, was introduced in JDK 1.1
  – In 1998, came JDK 1.2, which introduced Java IDL, a Java API for interoperability and integration with CORBA
  – In 1999, the RMI over IIOP standard extension to the Java platform was introduced for JDK 1.1.6 and 1.2
  – Now that RMI over IIOP is integrated into J2SE version 1.3 and higher
Basic CORBA

- **ORBs (Object Request Broker)**
  - To manage communication
  - Mediate messages between objects

- **Stubs (Client Side) and Skeletons (Server Sides)**
  - To implement the inter-process communication
  - Encode and decode the messages through the ORB

- **IIOP (Internet Inter-ORB Protocol)**
  - For TCP/IP networking between ORBs

- **IDL (Interface Definition Language)**
  - For the messages sent with CORBA.

- **Services (about 17 services, so far)**
  - Name Services, like the name registry, to find server objects

---

**Messages with an ORB**

![Diagram](http://example.com/diagram.png)

**Figure 1:** A request passing from client to object implementation.
Basic CORBA

Messages between ORBs

![Diagram of ORBs and Interoperability](image)

**Figure 2: Interoperability uses ORB-to-ORB communication**

Basic IDL

- **IDL**, the programming language of CORBA
- **IDL-to-Java Mapping** (see more on textbook table 15.1)

<table>
<thead>
<tr>
<th>IDL</th>
<th>JAVA EQUIVALENT</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>package</td>
<td>Groups together a set of interfaces or definition</td>
</tr>
<tr>
<td>interface</td>
<td>interface plus a helper class and holder class used by the server writer and the orb</td>
<td>Defines a set of methods that can be accessed by the client</td>
</tr>
<tr>
<td>constant</td>
<td>public static final</td>
<td>A non-mutable value</td>
</tr>
<tr>
<td>char, wchar</td>
<td>char</td>
<td>A character or “Wide” 2 byte character</td>
</tr>
<tr>
<td>string, wstring</td>
<td>java.lang.String</td>
<td>A string or “Wide” string</td>
</tr>
<tr>
<td>octet</td>
<td>byte</td>
<td>8 bits of information</td>
</tr>
<tr>
<td>long, unsigned long</td>
<td>int</td>
<td>A 4-byte integer</td>
</tr>
<tr>
<td>long long, unsigned long long</td>
<td>long</td>
<td>An 8-byte integer</td>
</tr>
<tr>
<td>float</td>
<td>float</td>
<td>A 4-byte floating point value</td>
</tr>
<tr>
<td>double</td>
<td>double</td>
<td>An 8-byte floating point value</td>
</tr>
</tbody>
</table>
Basic IDL

- Essential Java IDL API

<table>
<thead>
<tr>
<th>Package and API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>package org.omg.CORBA</td>
<td>provides the mapping of the OMG CORBA APIs to the Java</td>
</tr>
<tr>
<td>package org.omg.CosNaming</td>
<td>provides the naming service for Java IDL</td>
</tr>
<tr>
<td>package org.omg.PortableServer</td>
<td>provides classes and interfaces for making the server side of your applications portable across multivendor ORBs</td>
</tr>
<tr>
<td>package org.omg.PortableInterceptor</td>
<td>provides a mechanism to register ORB hooks through which ORB services can intercept the normal flow of execution of the ORB</td>
</tr>
<tr>
<td>package org.omg.DynamicAny</td>
<td>provides classes and interfaces to enable Any values to be dynamically interpreted (traversed) and constructed through DynAny objects</td>
</tr>
<tr>
<td>api org.omg.CORBA.ORB</td>
<td>provides APIs for the CORBA Object Request Broker features</td>
</tr>
</tbody>
</table>

* New packages in J2SE v1.4

Steps to write a basic IDL file:

1. Declare the CORBA IDL module
2. Declare the interface
3. Declare the operations
Basic IDL

A very simple IDL:

```idl
// c:\Hello\Hello.idl
module HelloApp
{
    interface Hello
    {
        string sayHello();
    };
};
```

This defines a module, called HelloApp, with a single interface, called Hello. The Hello interface defines a single operation called sayHello that returns a string.

---

Basic IDL

Server Side Program (continued):

```java
// c:\Hello\HelloServer.java
// The package containing our stubs.
import HelloApp.*;

// HelloServer will use the naming service.
import org.omg.CosNaming.*;

// The package containing special exceptions thrown by the name service.

// All CORBA applications need these classes.
import org.omg.CORBA.*;

public class HelloServer
{
    public static void main(String args[])
    {
        try
        {
            // Create and initialize the ORB
            ORB orb = ORB.init(args, null);
            // Create the servant and register it with the ORB
            HelloServant helloRef = new HelloServant();
            orb.connect(helloRef);
        }
    }
}
```

---
Basic IDL

Server Side Program:

```java
// Get the root naming context
org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
NamingContext ncRef = NamingContextHelper.narrow(objRef);

// Bind the object reference in naming
NameComponent nc = new NameComponent("Hello", "");
NameComponent path[] = {nc};
ncRef.rebind(path, helloRef);

// Wait for invocations from clients
java.lang.Object sync = new java.lang.Object();
synchronized(sync) {
    sync.wait();
}
} catch(Exception e) {
    System.err.println("ERROR: " + e);
e.printStackTrace(System.out);
}
```

class HelloServant extends _HelloImplBase
{
    public String sayHello()
    {
        return "Hello world!!
";
    }
}

Basic IDL

Client Side Program:

```java
import HelloApp.*; // The package containing our stubs.
import org.omg.CosNaming.*; // HelloClient will use the naming service.
import org.omg.CORBA.*; // All CORBA applications need these classes.

public class HelloClient
{
    public static void main(String args[])
    {
        try
        {
            // Create and initialize the ORB
            ORB orb = ORB.init(args, null);

            // Get the root naming context
            org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
            NamingContext ncRef = NamingContextHelper.narrow(objRef);

            // Resolve the object reference in naming
            NameComponent nc = new NameComponent("Hello", "");
            NameComponent path[] = {nc};
            Hello helloRef = HelloHelper.narrow(ncRef.resolve(path));

            // Call the Hello server object and print results
            String Hello = helloRef.sayHello();
            System.out.println(Hello);
        } catch(Exception e) {
            System.out.println("ERROR: " + e);
e.printStackTrace(System.out);
        }
    }
}
```
Basic IDL

- **JAVA IDL Development Process**
  
  1. Define the remote interface
     - `c:\Hello>edit Hello.idl`
  
  2. Compile the remote interface
     - `c:\Hello> idlj -fail Hello.idl`
  
  3. Implement the server
     - `c:\Hello>edit HelloServer.java`
     - `javac HelloServer.java HelloApp*.java`
  
  4. Implement the client
     - `c:\Hello> edit HelloClient.java`
     - `c:\Hello>javac HelloClient.java HelloApp*.java`
  
  5. Start the applications
     - `c:\Hello> start tnameserv -ORBInitialPort 1050`
     - `c:\Hello>start java HelloServer -ORBInitialPort 1050`
     - `c:\Hello>java HelloClient -ORBInitialPort 1050`
     - `c:\Hello> Hello world!!`

“Hello” Example: CORBA Communication in the same machine

```bash
$ cphuang@weblabpc3:~/opt/home/cphuang/public_html/CORBA
```

```bash
$ cphuang@weblabpc3 CORBA]$ ls
HelloServer.java
$cphuang@weblabpc3 CORBA]$ idlj -fail Hello.idl
$cphuang@weblabpc3 CORBA]$ ls
Hello.idl HelloApp.java HelloServer.java
$cphuang@weblabpc3 CORBA]$ javac HelloServer.java HelloApp/*.java
$cphuang@weblabpc3 CORBA]$ ls
Hello.idl HelloApp.java HelloServer.java HelloClient.java
$cphuang@weblabpc3 CORBA]$ ls
Hello.idl HelloApp.java HelloServer.java HelloClient.java
$cphuang@weblabpc3 CORBA]$ tnserv -ORBInitialPort 1050
```

```
12/3/01 IWS II 91.514 -CORBA
```

```
12/3/01 IWS II 91.514 -CORBA
```

```
Hello world!!
```

```
12/3/01 IWS II 91.514 -CORBA
```

```
12/3/01 IWS II 91.514 -CORBA
```

```
12/3/01 IWS II 91.514 -CORBA
```

```
12/3/01 IWS II 91.514 -CORBA
```
“Hello” Example: remote CORBA communication in 2 systems

```
import HelloApp.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
// Needed for the applet.
import java.awt.Graphics;
import java.awt.Font;
public class HelloApplet extends java.applet.Applet
{
    Font theFont = new Font("TimesRoman", Font.BOLD, 24);
    String message = " ";
    public void init()
    {
        // The same code as HelloClient.java here, except adding the followings
        // Call the Hello server object and print the results
        message = helloRef.sayHello();
    }
    public void paint(Graphics g)
    {
        g.setFont(theFont);
        g.drawString(message, 25, 50);
    }
}
```

“Hello” Example: Implement CORBA in a JAVA Applet

```
//HelloApplet.java
import HelloApp.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
// Needed for the applet.
import java.awt.Graphics;
import java.awt.Font;
public class HelloApplet extends java.applet.Applet
{
    Font theFont = new Font("TimesRoman", Font.BOLD, 24);
    String message = " ";
    public void init()
    {
        // The same code as HelloClient.java here, except adding the followings
        // Call the Hello server object and print the results
        message = helloRef.sayHello();
    }
    public void paint(Graphics g)
    {
        g.setFont(theFont);
        g.drawString(message, 25, 50);
    }
}
```
"Hello" Example: HTML for CORBA & Applet

```html
<html>
<head>
  <title>Getting Started with Java IDL: Running HelloApplet</title>
</head>
<body>
  <h1 ALIGN="CENTER">Running the Hello World Applet</h1>
  <p>
    If all goes well, the applet appears below:
  </p>
  <applet CODE="HelloApplet.class" COBJBASE="/" WIDTH=200 HEIGHT=100>
    <param name="org.omg.CORBA.ORBInitialPort" value="1000">
    <param name="org.omg.CORBA.ORBInitialPort" value="1000">
  </applet>
</body>
</html>
```

Note:
You may need Java Plug-in 1.3 for the web browser.
I use Netscape 6.2 in this case.

"Hello" Example: view CORBA result

Running the Hello World Applet

If all goes well, the applet appears below:

```
Hello world!
```

Note:
You may need Java Plug-in 1.3 for the web browser.
I use Netscape 6.2 in this case.
**“Simple” Example from Textbook, Chapter 15: Server-Side**

**Step 1:**
```bash
> tnameserv -ORBInitialPort 900
```

**Step 2:**
See the following figure

---

**“Simple” Example from Textbook, Chapter 15: Client-Side**

**Run Client:** SetValue, GetValue, RemoveValue

**Steps:** Follow the steps in this figure, (or see simpletest.bat in chapter 15).

---
“Session” Example from Textbook, chapter 16:

//SimpleSessionManager.java
import Sessions.*;
import java.util.*;
import org.omg.CORBA.*;
public class SimpleSessionManager extends _SessionManagerImplBase
{
    private Vector watchers;
    private Hashtable sessions;
    private ORB orb;
    .......
}

//SessionServer.java
import Sessions.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
public class SessionServer
{
    public static void main(String args[])
    {
        try
        {
            ORB orb = ORB.init(args, null);
            //Create and connect the registry object
            SimpleSessionManager manager = new SimpleSessionManager(orb);
            orb.connect(manager);
            .......
        } catch (Exception e) { ....... }
    }
}

Session Manager Design:

The First Client: SessionTest
C:\ETJava\16_sessionserver>java SessionTest -ORBInitialPort 900
Colors third = Bronze
Colors second = Silver
Colors first = Gold
Names Joe = cool
Names Master = Blaster
Names Mister = zero
Names Stephen = Ashbury
C:\ETJava\16_sessionserver>

The Second Client: WatchSession
C:\ETJava\16_sessionserver>java WatchSession -ORBInitialPort 900
Value in colors for first changed.
Value in colors for second changed.
Value in colors for third changed.
Value in names for Stephen changed.
Value in names for Minter changed.
Value in names for Master changed.
Value in names for Joe changed.
C:\ETJava\16_sessionserver>
Java/CORBA

Comparing Java/CORBA ORBs and Their Competition:

<table>
<thead>
<tr>
<th>Feature</th>
<th>CORBA/II OP</th>
<th>DCOM</th>
<th>RMI/RMP</th>
<th>HTTP/CGI</th>
<th>Servlet</th>
<th>Socket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstraction Level</td>
<td>★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★</td>
<td>★ ★ ★</td>
<td>★</td>
</tr>
<tr>
<td>Seamless Java Integration</td>
<td>★ ★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★</td>
</tr>
<tr>
<td>OS Platform support</td>
<td>★ ★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
</tr>
<tr>
<td>All-Java Implementation</td>
<td>★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
</tr>
<tr>
<td>Performance</td>
<td>★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ slow</td>
<td>★</td>
<td>★ ★ ★</td>
<td>★</td>
</tr>
<tr>
<td>URL-based naming</td>
<td>★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
</tr>
<tr>
<td>Distributed method</td>
<td>★ ★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★ ★</td>
<td>★ ★ ★</td>
<td>★</td>
</tr>
</tbody>
</table>


Summary

- CORBA examples have been shown in this presentation.
  - Including implement processes and results
- CORBA is a language-neutral specification
- IDL is used to define the interface between CORBA applications
- CORBA uses ORB to manage network communications
- Deciding Whether to Use RMI or IDL
  - 100% Pure Java vs. Support for Legacy Applications
  - Communication Protocols (JRMP vs. IIOP)
  - Objects by Value (RMI), Objects by Reference (for IDL)
References

- Stephen Asbury, Enterprise Linux at Work, J. Wiley, Chapter 4, 2000