

Java IDL

Design Patterns In Java

Bob Tarr

CORBA Overview

- CORBA - Common Object Request Broker Architecture
 - ⇒ Developed by the Object Management Group (OMG)
 - ⇒ Generic framework for distributed object applications
 - ⇒ Language and platform independent
 - ⇒ Object interfaces defined in an Interface Definition Language (IDL)
 - ⇒ An Object Request Broker (ORB) facilitates the client-server request/response action

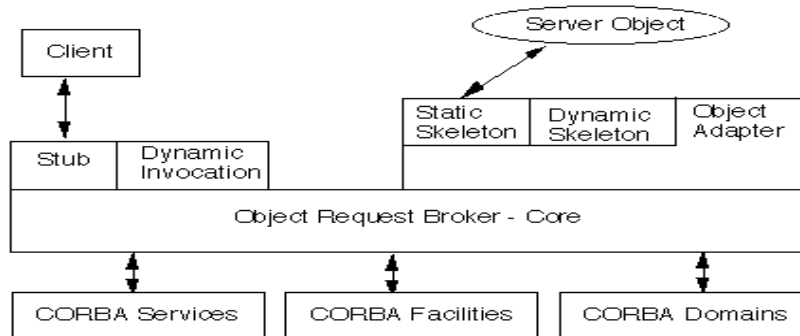
Design Patterns In Java

Java IDL
2

Bob Tarr

CORBA Overview

- CORBA Architecture



CORBA Overview (Continued)

- Object Request Broker (ORB)
 - ⇒ The Object Bus
 - ⇒ Static/Dynamic method invocations
 - ⇒ Supports method invocations of remote objects written in any supported language
- CORBA Interface Definition Language (IDL)
 - ⇒ Defines a remote interface
 - ⇒ Used by IDL compiler to generate stubs and skeletons
 - ⇒ Purely declarative; no implementation
 - ⇒ Syntax similar to C++
 - ⇒ Language bindings available for C, C++, Ada, Smalltalk, Cobol, Java
- Internet Inter-Orb Protocol (IIOP)
 - ⇒ Standard message formats and data representations for communications between ORBs using TCP/IP

JAVA IDL

- Java IDL
 - ⇒ Free from SUN as part of JDK 1.2
 - ⇒ Java ORB
 - ⇒ IDL Compiler
 - idltojava in Java 1.2 (available only as a separate download)
 - idlj in Java 1.3 (part of the standard Java 1.3 SDK)
 - ⇒ ORB API
- Java IDL Steps
 - ⇒ 1. Write the IDL interface file
 - ⇒ 2. Compile the IDL file to generate stubs and skeletons
 - ⇒ 3. Write and compile the remote object implementation
 - ⇒ 4. Write and compile the remote server
 - ⇒ 5. Write and compile the client

Java IDL Example

- This is the classic "Hello World" using Java IDL



Java IDL Example (Continued)

- First, define the interface using IDL. Here is the Hello.idl file:

```
module HelloApp {  
    interface Hello {  
        string sayHello();  
    };  
};
```

- Next, compile the IDL file to generate the stubs and skeletons:

```
idltojava Hello.idl
```

Java IDL Example (Continued)

- This generates the following files:
 - ⇒ `_HelloImplBase.java` - This abstract class is the server skeleton, providing basic CORBA functionality for the server. It implements the `Hello.java` interface. The server class `HelloServant` extends `_HelloImplBase`.
 - ⇒ `_HelloStub.java` - This class is the client stub, providing CORBA functionality for the client. It implements the `Hello.java` interface.
 - ⇒ `Hello.java` - This interface contains the Java version of our IDL interface. It contains the single method `sayHello`. The `Hello.java` interface extends `org.omg.CORBA.Object`, providing standard CORBA object functionality as well.
 - ⇒ `HelloHelper.java` - This final class provides auxiliary functionality, notably the narrow method required to cast CORBA object references to their proper types.
 - ⇒ `HelloHolder.java` - This final class holds a public instance member of type `Hello`. It provides operations for out and inout arguments, which CORBA has but which do not map easily to Java's semantics.

Java IDL Example (Continued)

- Note that the Java 1.3 idlj compiler generates a slightly different set of files with the _HelloImplBase.java file being replaced by a HelloOperations.java file
- Here is the generated Hello.java file:

```
/* Hello.java as generated by idltojava */
package HelloApp;
public interface Hello
    extends org.omg.CORBA.Object {
    String sayHello();
}
```

Java IDL Example (Continued)

- Here's the remote object implementation. All we have to do is extend the automatically generated implementation base and add the functionality that implements the required interface.

```
public class HelloServant extends _HelloImplBase {
    public String sayHello() {
        return "\nHello world!!\n";
    }
}
```

Java IDL Example (Continued)

- Here is the remote server, 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.
import org.omg.CosNaming.NamingContextPackage.*;

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

Java IDL Example (Continued)

```
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);

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

Java IDL Example (Continued)

```
// Bind the object reference in the root naming context
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);
}
}
}
```

Java IDL Example (Continued)

- And, finally, here is the client, HelloClient.java:

```
import HelloApp.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;

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);
```

Java IDL Example (Continued)

```
// 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);
}
}
}
```

Java IDL Example (Continued)

- Now let's try it!
 - ⇒ Start the Java IDL name server:
`tnameserv -ORBInitialPort 1050`
 - ⇒ Start the Hello server:
`java HelloServer -ORBInitialPort 1050`
 - ⇒ Run the Hello application client:
`java HelloClient -ORBInitialPort 1050`
 - ⇒ The client output!
`Hello world!!`