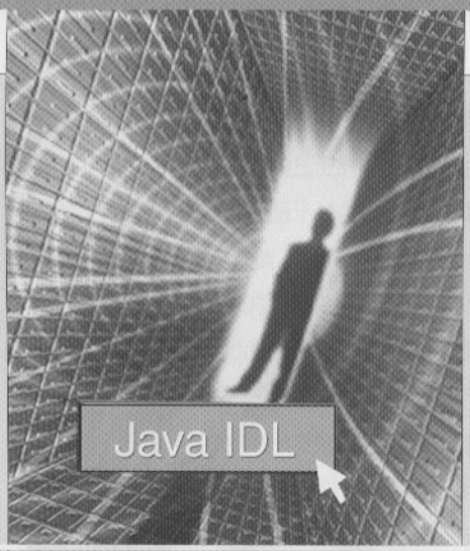

用 Java IDL 实现 CORBA 应用系统

Implementation of Application System Based on CORBA with Java IDL

倪良锁 赵毅梅 周 洲
（中央财经大学 信息系 100081）

摘 要：本文在简要介绍CORBA系统的组成和运行原理以及Java IDL的使用的基础上，通过一个简化了的基于CORBA的远程选课登记系统的实现来说明如何使用Java IDL来实现CORBA应用系统。

关键词：分布式系统 CORBA Java IDL

随着C/S结构的成熟，分布式应用系统得到了广泛的应用。CORBA规范清晰的定义了分布式系统中中间件的各个组成部件的功能和相互关系，借助于CORBA技术可以比较容易的实现分布式应用系统。本文先介绍CORBA技术，然后在此基础上通过一个实例来说明CORBA系统的实现。

1 CORBA 技术简介

1.1 CORBA 规范

CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）是由OMG（Object Management Group，对象管理组织）提出的面向对象的分布式系统的体系结构和技术规范，其核心是一套标准的接口定义语言IDL（Interface Definition Language，接口定义语言）、ORB和IIOP协议，用以支持分布式应用程序间的互操作以及独立于平台和编程语言的对象重用。

1.2 CORBA 系统的组成

CORBA系统的体系结构如图1所示：

（1）IDL语言，IDL文件和IDL编译器IDL语言是由OMG提出的一种接口定义语言，IDL语言有完整的语法规则和语义解释，从形式上看，它类似于C++语言。但它是一种描述性语言，由它编写的IDL文件，必须被映射成某种高级程序语言形式的文件，IDL编译器就是完成这项工作的映射工具。CORBA规范中已经定义了IDL语言到C，C++，Java等语言的映射规则。

IDL文件是用于描述客户程序和服务器程序交互的接口，客户程序将远程调用接口中定义的方法，服务器程序必须在对象实现中实现这些接口以及接口中定义的方法。

IDL文件经IDL编译器处理后将产生一系列文件，这些文件分别在客户程序和服务器程序中用作Stub和Skeleton，用于客户程序的静态调用和服务端的静态实现。

（2）Stub和Skeleton。Stub是客户端代理，它负责把客户程序的请求进行编码，传给客户端ORB，以及对客户端ORB传回的结果进行解释，返回给客户程序。

Skeleton是服务器端代理，它在服务端对来自BOA的请求进行解码，定位所请求的方法，执行该方法并把对象实现执行的结果发

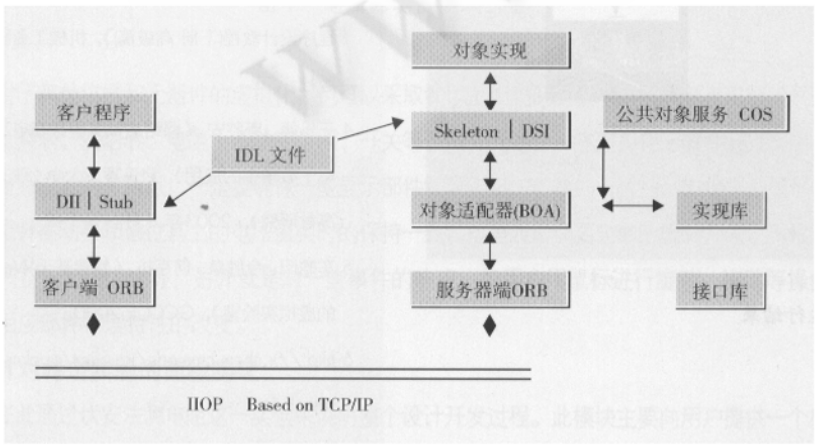

图 1

送给BOA。

(3) 对象适配器（OA，Object Adaptor）

对象适配器位于ORB与对象实现之间，负责对象实现的注册，对象引用的创建和解释，对象实现进程的激活和去活，以及客户请求的分发。

(4) 对象请求代理（ORB, Object Request Broker）

ORB提供了客户程序与对象实现之间透明通信的方法，它可以屏蔽对象实现的位置、实现方式、状态和通信机制等细节。

(5) 接口库(Interface Repository)和实现库(Implementation Repository)

接口库是用来存储、发布、管理相关对象的接口定义信息。ORB系统为了正确处理调用请求，必须掌握所处理的对象的接口信息。

实现库作用是存储了服务器端的对象实现的信息，供ORB/BOA定位和激活这些对象实现时去查询。

(6) 公共对象服务（COS，Common Object Service）

COS是由CORBA系统提供的一些独立于应用领域的基本服务，OMG的COSS规范定义了以下公共对象服务：名录服务（Naming Service）、事件服务（Event Service）、安全服务（Security Service）等。

(7) IIOP协议 (IIOP, Internet Inter-ORB Protocol)

CORBA 2.0规范中为不同厂商的ORB系统间的互操作制定了规范，它定义了GIOP，对数据编码、消息格式和传输协议作了规定。GIOP是一种抽象协议，在实现时必须映射到具体的某种运输层协议。CORBA支持GIOP到TCP/IP的映射，这种映射称为IIOP协议。

(8) 对象引用、客户程序和对象实现

对象引用（OR, Object Reference）是用来在ORB系统内标识一个对象实现的信息。客户程序通过访问目标对象实现的对象引用，就像引用本地对象一样来调用远程的目标对象实现的方法，客户程序仅需知道目标对象实现的结构，即对象的接口信息。对象实现则以多种形式来提供对象，并实现其中的方法。

1.3 CORBA 系统运行过程

当一个客户程序要调用某个远程对象实现的方法时，CORBA系统运行过程大致如下：

(1) 客户程序通过某种方式找到目标对象实现的对象引用；

(2) 如果目标对象实现有相应的Stub，则客户程序通过该Stub将请求编组；

(3) 当通过Stub，请求编组到达客户端ORB以后，客户端ORB借助于IIOP协议与服务端ORB交互，将请求编组传送到服务端ORB；

(4) 对象适配器接到来自服务端ORB的请求编组后，反编组，判断该请求的对象实现是否有Skeleton存在，如果有，则对象适配器通过Skeleton执行对象实现中的方法；

(5) 对象实现的方法执行完成以后，在本地保存结果，或将结果或异常将按请求的传递路径逆向返回给客户程序。

至此，一个完整的对象请求调用便完成了。

2 Java IDL的使用

Java IDL是Sun Microsystem为编程者提供的一个免费的ORB系统，它包含于JDK1.3及后续版本中。Java IDL 提供了IDL编译器idlj.exe，。下面简要介绍IDL编译器idlj.exe的使用（以IDL文件My.idl为例）：

(1) idlj My.idl

(2) idlj —fclient My.idl

这两条命令结果相同，将产生IDL文件My.idl到Java文件的映射，但只产生客户端的Stub文件。

(3) idlj —fserver My.idl

这条命令将产生服务端的Skeleton文件。

(4) idlj —fclient —fserver My.idl

(5) idlj —fall My.idl

这两条命令将产生客户端和服务端的Java文件。使用第5条命令将产生如下6个文件：

① _MyStub.java 这个文件是客户端的Stub；

② _MyImplBase.java 这个文件是服务端的Skeleton；

③ My.java 这是个接口文件，它是IDL文件中所定义的接口的Java语言版本；

④ MyHelper.java 这个文件提供一些辅助功能；

⑤ MyHolder.java IDL语言中有输出型（out）或输入型（inout）参数，而Java中没对应的参数类型，MyHolder.java则提供关于这些类型参数的操作。

⑥ MyOperations.java IDL文件中定义的所有方法将映射到这个文件中。

3 用Java IDL实现的一个基于CORBA的应用系统实例

在介绍CORBA系统和Java IDL使用的基础上，本文以一个简化的某高校继续教育学院网上选课注册系统为例，介绍怎样使用Java IDL来实现基于ORBA的应用系统。因为学员分布于全国各地，故客户端采用基于Java Applet方式来实现。下面分5步来完成该系统：

(1) 写IDL接口文件 registry.idl

```
module registryApp
{
    interface registry
    {string regPersonalInfo(in string name
, in string major);
    }
}
```

(2) 用idlj.exe编译registry.idl

idlj —fall registry.idl

(3) 编写客户端程序regClient.java，并编

译之

```
//The package containing our stubs.
import registryApp.*;
//regclient will use 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.*;
import java.awt.*
public class regClient extends java.applet.Applet implements ActionListener
{ string name , message, major;
Button b1;
TextField tf1,tf2;
public void init ( )
{Label t1,t2;
setBackground(Color.white);
setLayout(new FlowLayout(FlowLayout.LEFT));
t1=new Label("姓名");
tf1=new TextField("你的姓名", 10);
t2=new Label("专业");
tf2=new TextField("你的专业",8);
b1=new Button("确定");
b1.addActionListener(this);
add(t1);
add(tf1);
add(t2);
add(tf2);
add(b1);
}
public void actionPerformed(Action e)
{ name=tf1.text;
major= tf2.text;
try
{ //create and initialize the ORB
ORB orbclient= ORB.init(this , null);
//resolve the name service to get the root naming context
org.omg.CORBA.Object objRef = orbclient.resolve_initial_refences("NameService");
NamingContext ncRef = NamingContextHelper.narrow(objRef);
//resolve the object refence in naming context
NameComponent nc =new NameComponent("registry", " ");
NameComponent path[ ] = {nc};
registry regRef = registryHelper.narrow(ncRef.resolve(path));
//call the server object
message = regRef.regPersonalInfo(name , major);
//print the result
System.out..println(message +"\n");
}
catch (Exception e) {
System.out..println ( "regclient Exception : " +e ) ;
e.printStackTrace (System.out);
}
}
}
```

编写html 文件，将regClient.class嵌入到html文件中

```
<html>
<head><title>选课登记系统</title></head>
<body>
<applet code="regClient.class" width="480" height="200"></applet>
</body>
</html>
```

(4) 编写服务端程序 regServer.java，并编译之（代码略）

(5) 部署应用系统

4 结束语

使用CORBA可以很容易的实现分布式应用系统，但是CORBA也有它的缺点，比如，它使用IIOP协议，可能不能穿过防火墙。即使如此，CORBA技术仍然是实现分布式系统比较成熟的技术，在企业信息系统中有着广泛的使用。

参 考 文 献

1 花汪芸，CORBA技术及其应用[M]，东南大学出版社，1999。

2 刘军万 等，面向对象的分布式CORBA技术及其实现[J],计算机工程与应用，2002(10)。

3 Vogel A, Duddy K Java Programming with CORBA[M],NewYork:JohnWiley&Sons,Inc, 1998。