# 一 IEC框架介绍

Ice 是一种面向对象的中间件平台。从根本上说，这意味着Ice 为构建面向对象的客户－服务器应用提供了工具、API和库支持。Ice 应用适合在异种环境中使用：客户和服务器可以用不同的编程语言编写，可以运行在不同的操作系统和机器架构上，并且可以使用多种网络技术进行通信。无论部署环境如何，这些应用的源码都是可移植的。

什么是ICE?   
   ICE(internet communications engine)是适用于异种环境的面向对象中间件平台。   
   那么什么是中间件呢?   
   比较流行的定义是:中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件位于客户机/服务 器的操作系统之上，管理计算资源和网络通讯。   从中间件的定义可以看出，中间件是一类软件，而非一种软件；中间件不仅仅实现互连，还要实现应用之间的 互操作；中间件是基于分布式处理的软件，定义中特别强调了其网络通讯功能.    
2.ICE的一些概念   
服务器/客户端(server/client):这个的定义与一般的定义相同,主动的一方被认为是client   
ICE对象:跟OOP中的对象类似,不同之处在于,在分布式的环境中,同一个ICE对象在不同的地址空间中都可能存在着.ICE对象也提供了一组接口(facets).ICE对象还有一个特殊的接口:主接口.   
代理(proxies):是ICE对象引用,代理是在客户地址空间,客户对ICE对象的操作就是通过代理来进行的.代理封装了完成:ICE对象的寻址(包括服务器的寻址),激活ICE对象,传入参数,等待执行并返回执行结果   
servant:在服务器上的执行体,ICE对象对服务器的操作就是通过调用servant.   
"最多一次"原则:一次对目的的访问,只会执行一次(并不排除出错重试)   
ICE的5个服务   
Summary这里介绍了ICE的五个服务   
        Ice为分布式开发提供了技术完善的客户-服务平台。实际上，现实的应用不仅仅只需要具备远程通讯能力，通常的，还需要随需启动服务，把代理分布到客户端，分发异步事件，配置应用，发布补丁等等。   
        Ice自带了一些服务，这些服务具备了上面所说的特征和一些其它的特征。这些服务实现以服务器的形式实现并向你的应用提供服务。他们分别是：   
1、IcePack   
         IcePack是Ice的定位服务。当使用间接绑定时，用来将符号化的适配器名称转换为协议-地址对。除了定位服务之外，IcePack还提供了如下的服务：   
IcePack允许你注册一个自动启动的服务：即当客户端进行请求时，服务器不需要处于运行状态，只要第一个客户端进行请求时，服务会自动启动。   
IcePack支持脚本描述部署，可以轻易的配置包含了若干个服务的复杂的应用   
IcePack提供了简单的对象查找服务，允许客户端获取他们感兴趣的对象代理。   
2、IceBox   
        IceBox是一个简单的应用服务器，它可以协调多个应用组件启动和停止。   
        应用组件可以用动态链接库的形式发布而不是一个进程。这就减轻了系统的负载。例如，你可以在一个JVM中运行若干个应用组件而不是有多个进程，每一个进程都有自己的JVM。   
3、IceStorm   
        IceStorm是一个发布-订阅服务，它减除了客户端和服务器的耦合度。本质上说，IceStorm作为一个事件分发的交换机运行。发布者将事件发给服 务，IceStorm按照顺序将事件传递给订阅者。使用这种方法，一个事件发布者就可以把一个事件发布给多个订阅者。事件按照主题分类，订阅者可以指定他 们感兴趣的主题。只有订阅者感兴趣的主题才会发送给订阅者。服务允许指定服务的质量，从而允许应用在可伸缩性和性能之间进行适当的折中。   
        如果你需要将信息发布到大量的应用组建，那么IceStorm是一个不错的选择。（一个很典型的例子就是股票报价的应用）。IceStorm减除了信息的 发布者和订阅者之间的耦合，同时也能重新发布已经发布的信息。另外，IceStorm可以作为联合服务运行，即多个服务的实例可以运行在不同的机器上，从 而降低了CPU的负载。   
4、IcePatch   
        IcePatch是一个软件补丁服务。它允许你轻松的把软件的更新发布给客户。客户连接到IcePatch然后请求更新一个特定的应用。服务就自动检查客 户软件的版本然后下载需要更新的组件。而这些下载的组件都是放在一个压缩包里的，从而减少了带宽的占用。软件补丁也可以通过结合Glacier服务发布， 这样可以让只有经过授权的客户才能下载软件更新。   
5、Glacier   
        Glacier是Ice防火墙服务：它允许客户和服务器通过防火墙安全的通讯。客户-服务器的通讯通过使用公钥认证完全加密，并且通讯是双向的。Glacier提供了相互认证和安全的会话管理支持。

# 二 环境搭建

 1 下载最新安装包; <http://www.zeroc.com/download.html> 根据操作系统和需要选择,我这里安装的是ice-3.4.2

 2 安装之后的Ice相关路径:   
   slice2cpp,slice2java在/bin/下   
   Ice.jar 存储于/lib/java2/下   
   相关的Ice的库存储于/lib下.

# 三 Demo举例

需求：做一个员工信息增加、删除、修改、查询操作的功能。 客户端发送相应参数连接服务器进行以上四种操作，服务端接收请求并完成对应操作，并返回操作结果。

员工信息表中包含：员工编号、员工名称、密码、部门编码、描述。

1.创建ice文件:    
数据模型model定义如下：

#ifndef \_MODEL

#define \_MODEL

module com

{

module hh

{

module demo

{

module emp

{

module model

{

/\*\*自定义Map类型\*\*/

dictionary<string, string> CustomMap;

/\*\* 消息结构 \*\*/

["java:getset"]

struct Employee {

/\*\*工号\*\*/

int empId;

/\*\*姓名\*\*/

string empName;

/\*\* 登录密码 \*\*/

string password;

/\*\* 部门编码 \*\*/

string deptCode;

/\*\* 描述 \*\*/

string description;

};

/\*\*定义整型数组\*\*/

sequence<Employee> EmployeeArray;

};

};

};

};

};

#endif

业务操对象service定义如下：

#ifndef \_GENERATED

#define \_GENERATED

#include <model.ice>

module com

{

module hh

{

module demo

{

module emp

{

module service

{

interface EmployeeServiceIce

{

/\*\*

\* 按条件查询员工信息

\* @param message 消息内容

\* @return true 成功 false 失败

\*/

model::EmployeeArray getEmployees(model::CustomMap params);

/\*\*

\* 添加用户

\* @param employee 员工对象

\* @return true 成功 false 失败

\*/

string addEmployees(model::Employee employee);

/\*\*

\* 修改用户

\* @param employee 员工对象

\* @return true 成功 false 失败

\*/

string updateEmployees(model::Employee employee);

/\*\*

\* 删除用户

\* @param employee 员工对象

\* @return true 成功 false 失败

\*/

string deleteEmployees(model::Employee employee);

};

};

};

};

};

};

#endif

2.dos环境下执行,(前提：将ICE安装目录配置在path中)

 cd E:/workspace/ICETest/slice

 E:/Ice-3.3.0/bin/slice2java -I. --output-dir=../src \*.ice //生产代码

编译完成后，会在output-dir指定的目录下生成相应的java文件。将这些文件引入到创建的java工程中。接下来需要使用到。

3编写服务器端代码：

**public** **class** IceDemoServer {

**public** **static** **void** main(String[] args) {

**int** status = 0;

Communicator ic = **null**;

**try** {

ic = Ice.Util.*initialize*(args);

Ice.ObjectAdapter adapter = ic.createObjectAdapterWithEndpoints(

"demoAdapter", "default -h \*");

EmployeeServiceIce employeeService = (EmployeeServiceIce) BeanUtils.*getBean*("employeeService");

adapter.add(employeeService, ic.stringToIdentity("employeeService"));

adapter.activate();

ic.waitForShutdown();

} **catch** (Ice.LocalException e) {

e.printStackTrace();

status = 1;

} **catch** (Exception e) {

System.*err*.println(e.getMessage());

status = 1;

}

**if** (ic != **null**) {

**try** {

ic.destroy();

} **catch** (Exception e) {

System.*err*.println(e.getMessage());

status = 1;

}

}

System.*exit*(status);

}

}

4客户端代码

**public** **class** IceClient {

**public** **static** **void** main(String[] args) {

**int** status = 0;

Communicator ic = **null**;

**try** {

**long** startTime = System.*currentTimeMillis*();

String str = String.*format*("%s:%s -h %s -p %s",

"IceTestServiceGrid/Locator", "tcp", "localhost", "10000");

InitializationData localInitializationData = **new** InitializationData();

localInitializationData.properties = Util.*createProperties*();

localInitializationData.properties.setProperty(

"Ice.Default.Locator", str);

ic = Util.*initialize*(localInitializationData);

EmployeeServiceIcePrx employeeService =EmployeeServiceIcePrxHelper

.*checkedCast*(ic.stringToProxy("employeeService"));

**if** (employeeService == **null** )

**throw** **new** Error("Invalid proxy");

Map<String, String> map = **new** HashMap<String, String>();

//插入需要的业务操作代码。

} **catch** (Ice.LocalException e) {

e.printStackTrace();

status = 1;

} **catch** (Exception e) {

System.*err*.println(e.getMessage());

status = 1;

}

**if** (ic != **null**) {

**try** {

ic.destroy();

} **catch** (Exception e) {

System.*err*.println(e.getMessage());

status = 1;

}

}

System.*exit*(status);

}

}

5。创建服务端配置文件

application.xml ：

<icegrid>

<application name="icetest">

<server-template id="icetest-server-template">

<parameter name="index" />

<server id="rescenter\_${index}" exe="F:\Program Files\Java\jdk1.6.0\_18\bin\java" activation="on-demand">

<!---server -Xms128m -Xmx256m -d64 -XX:PermSize=128m-->

<option>-classpath</option>

<option>F:\workspace\ICEDemoServer\bin</option>

<option>-Djava.ext.dirs=F:\workspace\ICEDemoServer\lib</option>

<option>-server</option>

<option>com.hh.demo.emp.IceDemoServer</option>

<adapter name="demoAdapter" endpoints="tcp" replica-group="ReplicatedTestAdapter" />

<property name="Ice.ThreadPool.Server.SizeMax" value="3000" />

</server>

</server-template>

<replica-group id="ReplicatedTestAdapter">

<load-balancing type="adaptive" load-sample="1" n-replicas="1" />

<object identity="employeeService" type="::com::hh::demo::emp::service::impl::EmployeeServiceImpl" />

</replica-group>

<node name="node1">

<server-instance template="icetest-server-template" index="1" />

</node>

</application>

</icegrid>

config-ice.grid：

IceGrid.InstanceName=IceTestServiceGrid

#

# The IceGrid locator proxy.

#

Ice.Default.Locator=IceTestServiceGrid/Locator:tcp -p 10000

#

# IceGrid registry configuration.

#

IceGrid.Registry.Client.Endpoints=tcp -p 10000

IceGrid.Registry.Server.Endpoints=tcp

IceGrid.Registry.Internal.Endpoints=tcp

IceGrid.Registry.Data=db/rescenter\_registry

IceGrid.Registry.PermissionsVerifier=IceTestServiceGrid/NullPermissionsVerifier

IceGrid.Registry.AdminPermissionsVerifier=IceTestServiceGrid/NullPermissionsVerifier

IceGrid.Registry.SSLPermissionsVerifier=IceTestServiceGrid/NullSSLPermissionsVerifier

IceGrid.Registry.AdminSSLPermissionsVerifier=IceTestServiceGrid/NullSSLPermissionsVerifier

#

# Dummy username and password for icegridadmin.

#

IceGridAdmin.Username=foo

IceGridAdmin.Password=bar

#set server active Connection Managerment

Ice.ACM.Server=60

#

# IceGrid node configuration.

#

IceGrid.Node.Name=node1

IceGrid.Node.Endpoints=tcp

IceGrid.Node.Data=db/node1

IceGrid.Node.CollocateRegistry=1

IceGrid.Node.Trace.Activator=1

IceGrid.Node.Trace.Adapter=2

IceGrid.Node.Trace.Server=3

在命令行中输入：

icegridnode --Ice.Config=config-ice.grid --deploy application.xml

pause

服务端程序就发布起来了。

运行客户端程序，就可以调用服务端程序。