JBuilderX+WebLogic Server 实现 JNDI

说明：

本人开发环境

硬件：P4 2.66G CPU/80G HDD/512M DDR

软件：Windows 2000 Adv Server/Bea Weblogic Platform 8.1 sp2/JBuilderX 10.0.205/IE 6.0.2600

运行通过。

JNDI(Java Naming and Directory Interface,JNDI) **是** J2EE **提供的命名和目录服务，该服务在分布式应用程序中是不可缺少的，它不仅提供了方便，更主要的是提供了一层间接引用。一下就这个问题进行简要讨论。**

JNDI 介绍

命名服务用来确定网络中可以访问的对象，在一个命名服务中，把一个名称和对象绑定在一起，并且可以通过给定的名称找到对应的对象。

JNDI 命名和目录接口为 Java **程序提供了命名服务。在** J2EE **规范中，JNDI 与具体的命名服务的实现无关，可以使用 JNDI 获得对现有命名服务、新创建的命名服务的访问，可以通过** JNDI **的标准服务提供商接口(SPI)去集成已有的命名服务。**

应用程序通过 JNDI API **访问 JNDI 命名服务。**

一个使用 JNDI **命名服务的接口可以分为三层：**Java **应用程序、**JNDI **命名管理器，以及不同的命名服务。对于使用命名服务的客户端而言，可以通过统一的接口访问不同的命名服务，那就是** JNDI**，从图** WSJ-1 **可以看出** JNDI **命名服务把** RMI、CORBA、LDAP **等现有命名服务进行封装，对用户提供统一接口，简化了客户端使用命名服务的复杂度。**

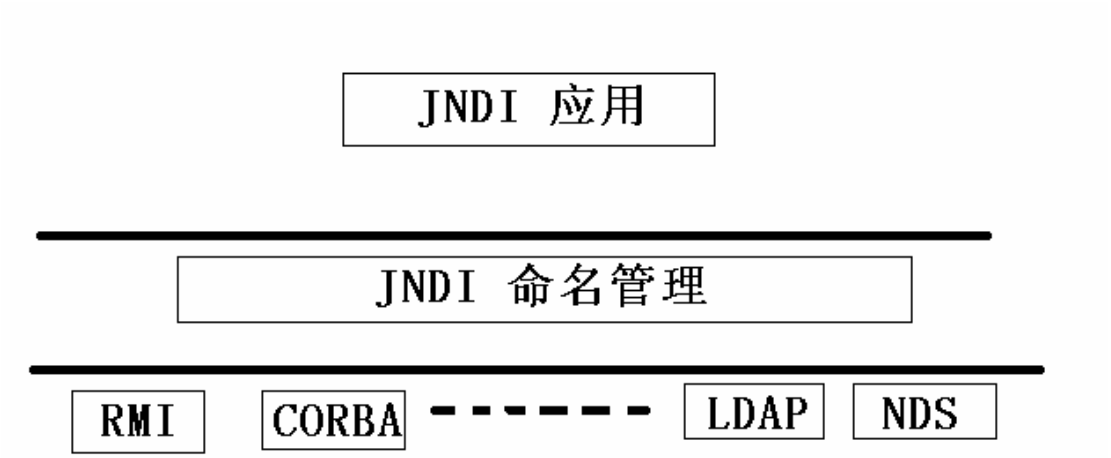


图 WSJ-1 JNDI 命名服务

JNDI 接口

JNDI API 包含了 4 个包：

- Javax.naming 包含访问命名服务的类与接口
- Javax.naming.event 包含在命名服务中实现事件通知机制的类和接口
- Javax.naming.ldap 包含支持 LDAP v3 扩展和控制的类与接口
- JNDI SPI 仅包含 javax.naming.spi,可以使用 JNDI SPI 集成不同的目录服务

下面对 JNDI 几个比较重要的概念进行介绍。

1、Contexts

Contexts 是指命名上下文的核心接口。Contexts 定义了命名服务的基本操作，例如增加对名称－对象绑定、根据名称查找对象，列出绑定对象、解除绑定，以及创建和删除子上下文等操作。下面是一段示例代码。

```
public interface Context{
public Object lookup(Name name) throws NamingException;
public void bind(Name name,Object obj) throws NamingException;
public void rebind(Name name,Object obj) throws NamingException;
public void unbind(Name name) throws NamingException;
public void rename(Name old,Name new) throws NamingException;
public void NamingEnumeration listBinding(Name name)
            throws NamingException;
```

…

public Context createSubcontext(Name name) throws NamingException;

public void destorySubcontext(Name name) throws NamingException;

…

}

2、The Initial Context

InitialContext **实现了** Context **接口，一般关于** JNDI **的操作都对类** InitialContext **操作。以下是简单的示例。**

Public class InitialContext implements Context{

Public InitialContext()…;

…

}

3、Bindings

绑定是指把一个对象与一个名称联系在一起，这样就可以提供从名称到对象的查询。

绑定到 WebLogic Server JNDI Tree

在 WebLogic **控制台中可以查看已经绑定的对象。**

首先，启动 WebLogic Server **，然后在浏览器中输入** http://localhost:7001/console **，登录后在左边的目录树中用鼠标右键单击** mydomain\servers\wellserver(wellserver **是我的** WebLogic server **名称)，在弹出的快捷菜单中选择**"View JNDI Tree"**，就可以查看** WebLogic **服务器提供的目录服务。如图** WSJ-2**、图** WSJ-3 **所示。**

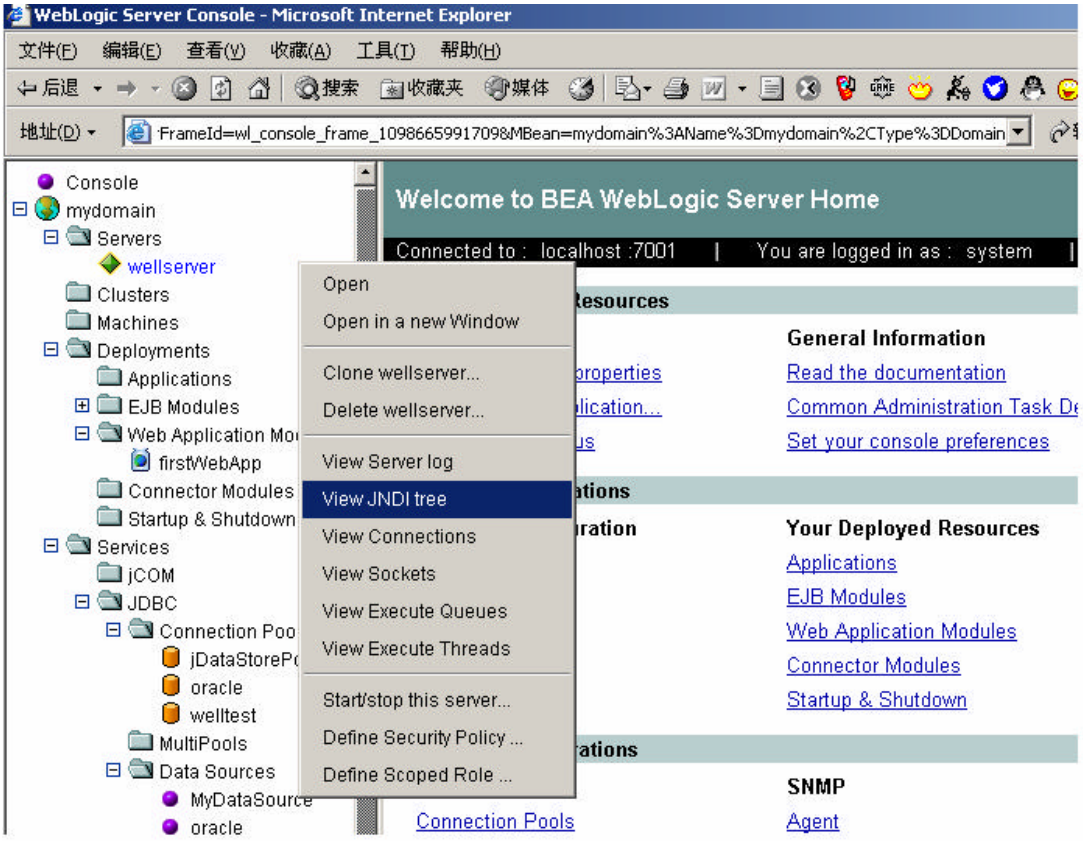


图 WSJ-2 查看WebLogic JNDI Tree

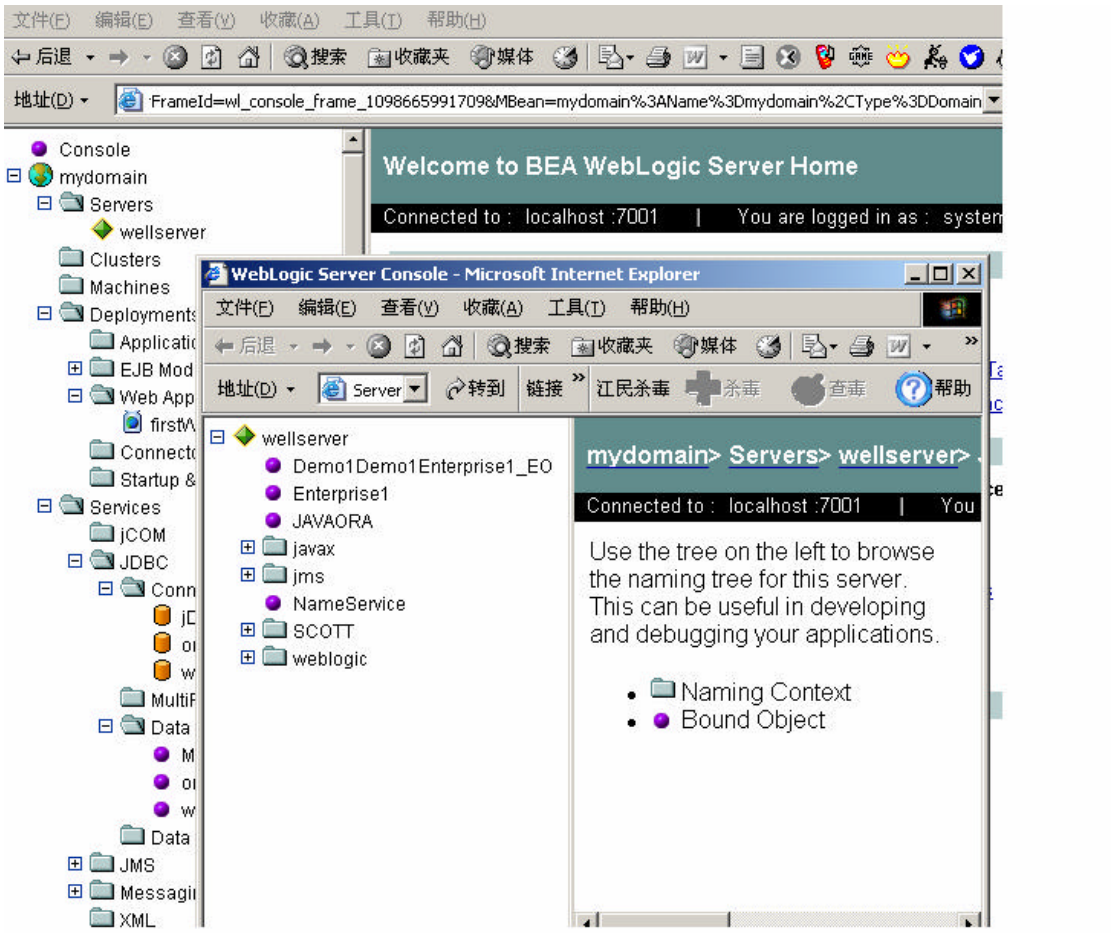


图 WSJ-3 查看WebLoigc JNDI Tree

JNDI 开发

开发WebLogic Server**的**JNDI**程序非常类似于开发** RMI**程序。开发一个** JNDI **程序一般包括以下几个步骤。**

1、**绑定对象**

为初始上下文 InitialContext **建立** JNDI **属性，如下所示：**

```
Hashtable ht = new Hashtable();
ht.put(Context.INITIAL_CONTEXT_FACTORY,
        “weblogic.jndi.WLInitialContextFactory”);
ht.put(Context.PROVIDER_URL,”t3://localhost:7001”);
```

以上代码的功能是将两个 WebLogic Server **的** JNDI **属性放入一个** HashTable **中，注意要将** Context.PROVIDER_URL **的值设定为自己的** WebLogic Server **地址和端口号。** Context.INITIAL_CONTEXT_FACTORY **用来指定工厂，**

Context.PROVIDER_URL 用来指定 URL。

使用该上下文在 WebLogic Server 名字空间中绑定对象：

```
try{
  //建立初始上下文
 Context ctx = new InitialContext(ht);
  //将参数 object 以 name 指定的名字绑定到 WebLogic Server 的名字空间
  ctx.rebind(name,object);
}
```

ctx.rebind(name,object)把 object 与名称 name 绑定在一起。name 是 String 类型，而 object 可以是任意的 Java 类型。

2、查询对象

为初始上下文 InitialContext 建立 JNDI 属性，使用该上下文就可以在 WebLogic Server 名字空间查询已经注册的对象，如下所示：

```
ctx = new InitialContext(ht);
Object object = ctx.lookup(name);
return object;
```

以上代码中的 Object object ＝ ctx.lookup(name)语句使用 JNDI 查询方法 lookup 查询绑定名称为 name(自己定义的 name)的对象。

完整示例：

1、在 JBuilderX 中新建一个工程 jndi。如图 WSJ-4 所示：

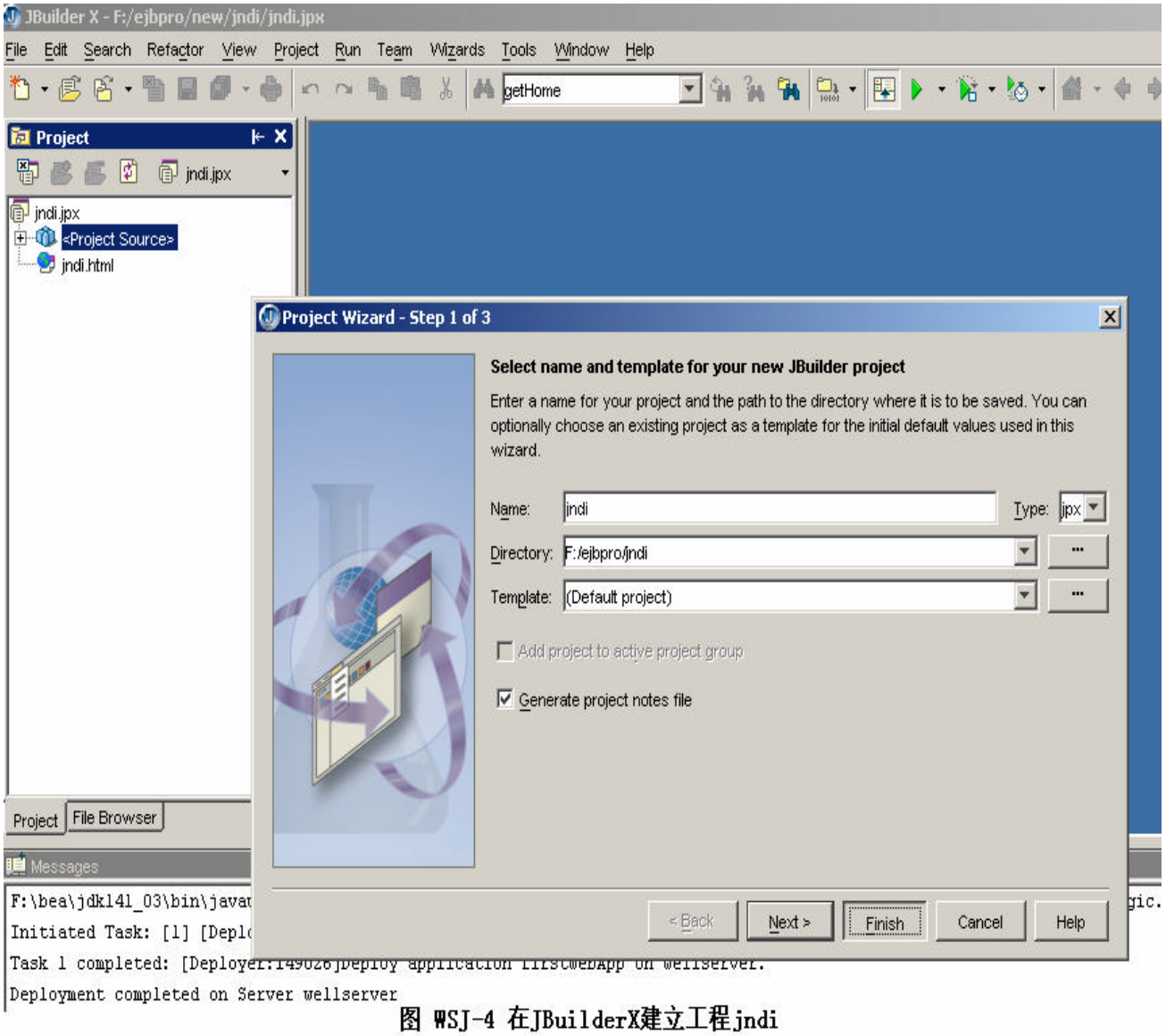


图 WSJ-4 在JBuilderX建立工程jndi

2、**在左边的** project **面板上，鼠标右键点击**<project sources>，**在弹出的快捷菜单中选择** new→class ，**在弹出的** new class **面板中写入类名：**JNDI，**如图** WSJ-5、**图** WSJ－6 **所示：**

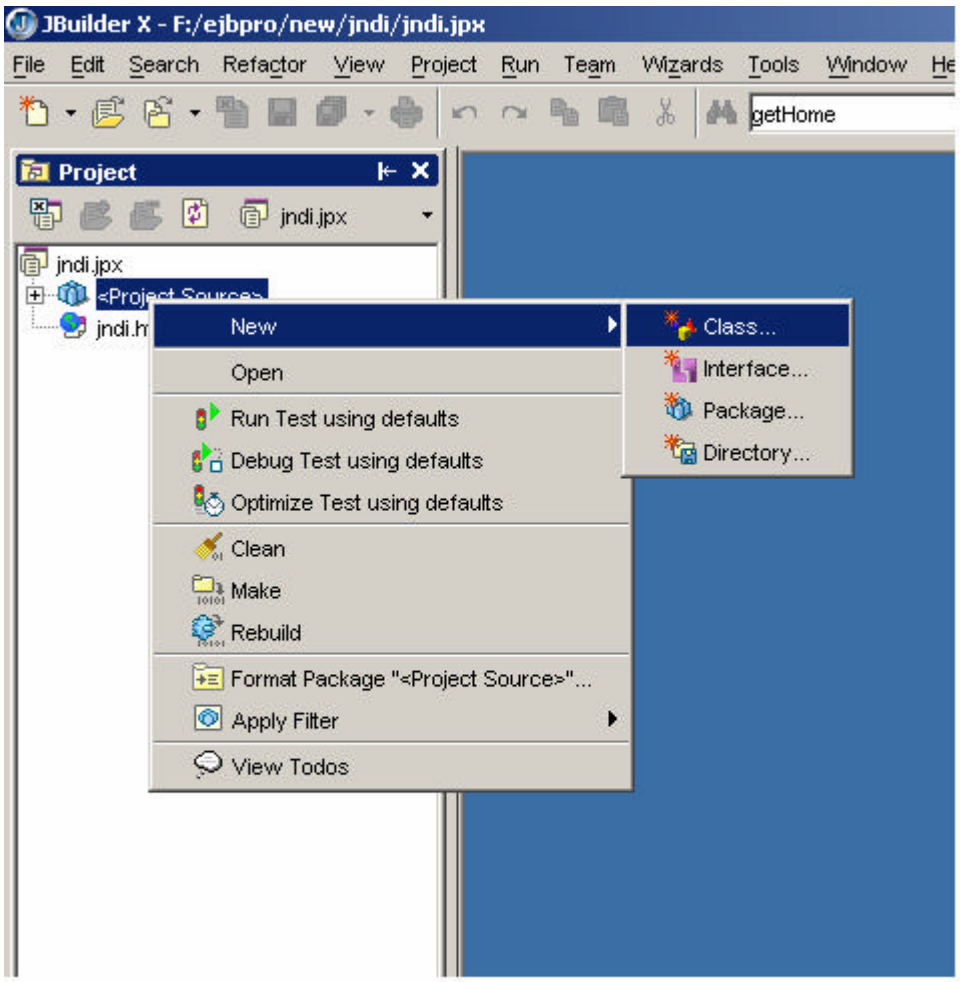


图 WSJ－5 建立一个 new class

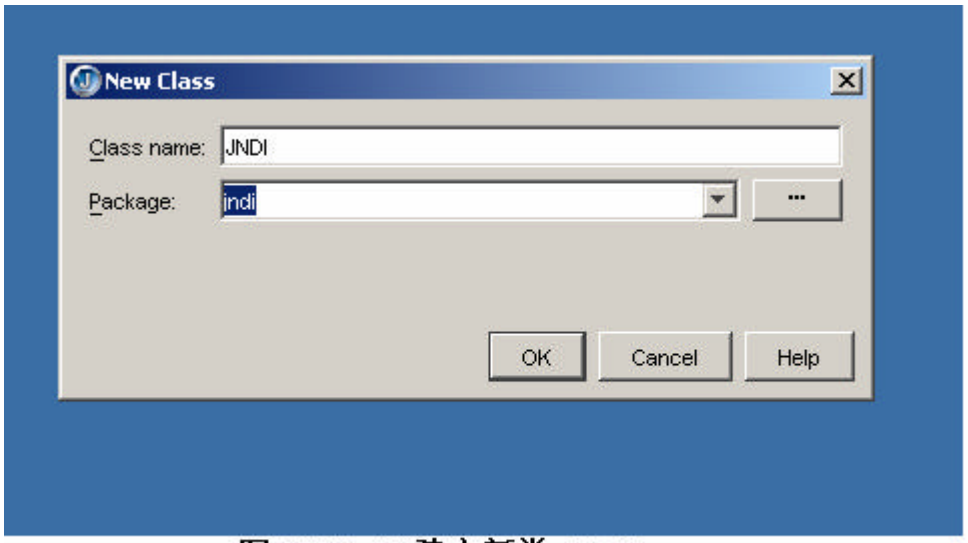


图 WSJ－6 建立新类 JNDI

3、**以下是全部代码。**

代码中定义了两个方法，bind **方法用来绑定一个对象到** WebLogic Server **的名字空间，**lookUp **方法用来通过** JNDI **查找指定名字的对象。最后是测试这两个方法的** main **方法。**

```
package jndi;

import javax.naming.*;
import java.util.Hashtable;

class JNDI{
 static Context ctx = null;
 public JNDI(){
  }
 //将对象 object 绑定到 WebLogic Server 的名字服务器中
 public static void bind(String name,String object){
  Hashtable ht = new Hashtable();
  String url = "t3://localhost:7001";

ht.put(Context.INITIAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFa
ctory");
    ht.put(Context.PROVIDER_URL,url);
    try{
     ctx = new InitialContext(ht);
      ctx.rebind(name,object);
    }
     catch(NamingException e){
       System.out.println(e);
      }
    finally{
     try{
```

```
        ctx.close();
       }
      catch(Exception e){
        e.printStackTrace();
       }

    }
  }
  //通过 JNDI 查询指定的对象
  public static Object lookUp(String name){
   Hashtable ht = new Hashtable();
   String url = "t3://localhost:7001";

ht.put(Context.INITIAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFa
ctory");
    ht.put(Context.PROVIDER_URL,url);
    try{
     ctx = new InitialContext(ht);
     Object object = ctx.lookup(name);
      return object;
    }
   catch(NamingException e){
      System.out.println(e);
    }
    finally{
      try{
        ctx.close();
       }
      catch(Exception e){
        System.out.println(e);
```

```
    }
   }
   return null;
  }
 public static void main(String args[]){
  String arg = args[0];
   if(arg.equals("bind")){
    System.out.println("bind begin ...");
    String bookName="WebLogic introduction";
     bind("bookname",bookName);
    System.out.println("bind end");
   }
   if(arg.equals("lookup")){
    System.out.println("lookup begin ...");
    String bookName;
    bookName = (String)lookUp("bookname");
    System.out.println("bookname is :  " + bookName);
    System.out.println("look up end.");
   }
  }

}
```

4、 **在** JBuilderX **中设置运行项以指定运行参数**

单击 JBuilderX **中的** Run→Configurations **选项，如图** WSJ－7 **所示；在弹出的** Runtime Configuration **面板中点击** new **按钮，如图** WSJ－8 **所示。**

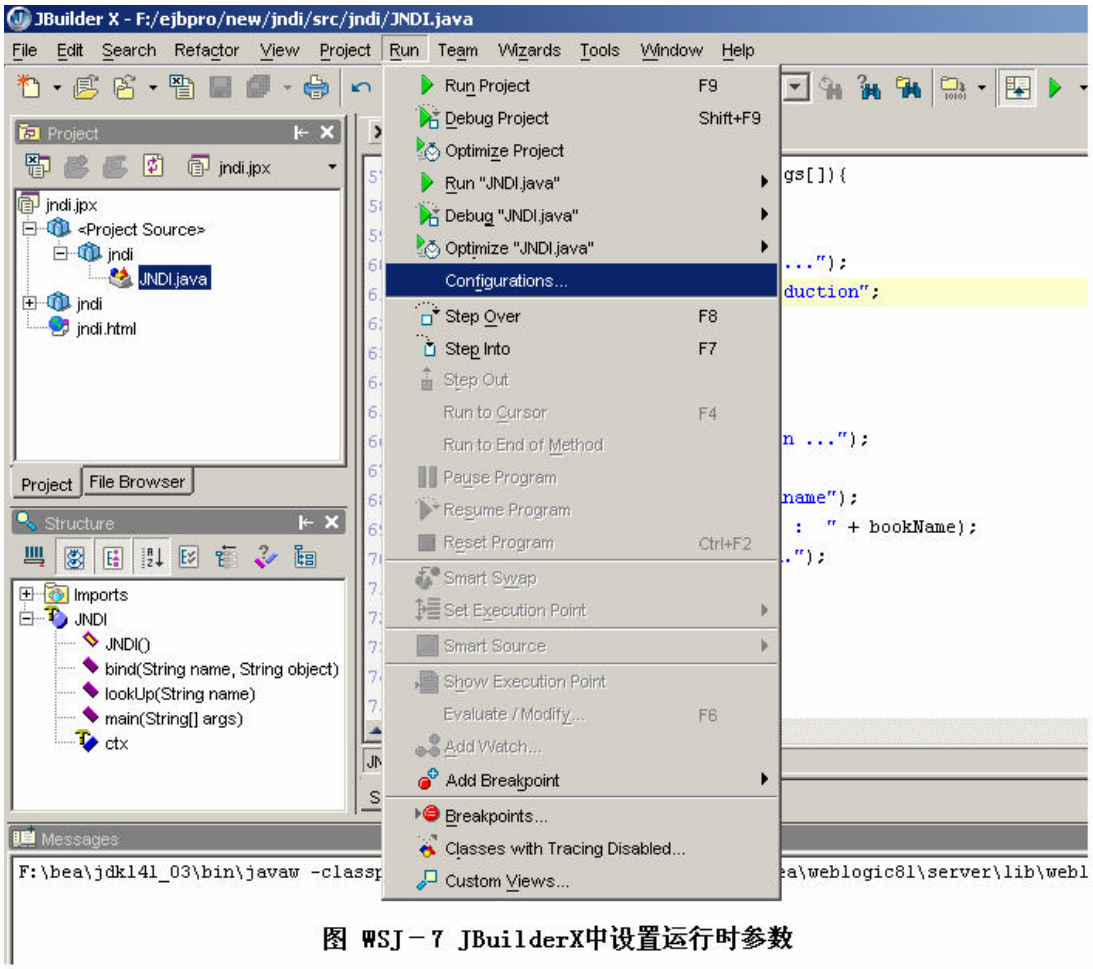


图 WSJ－7 JBuilderX中设置运行时参数

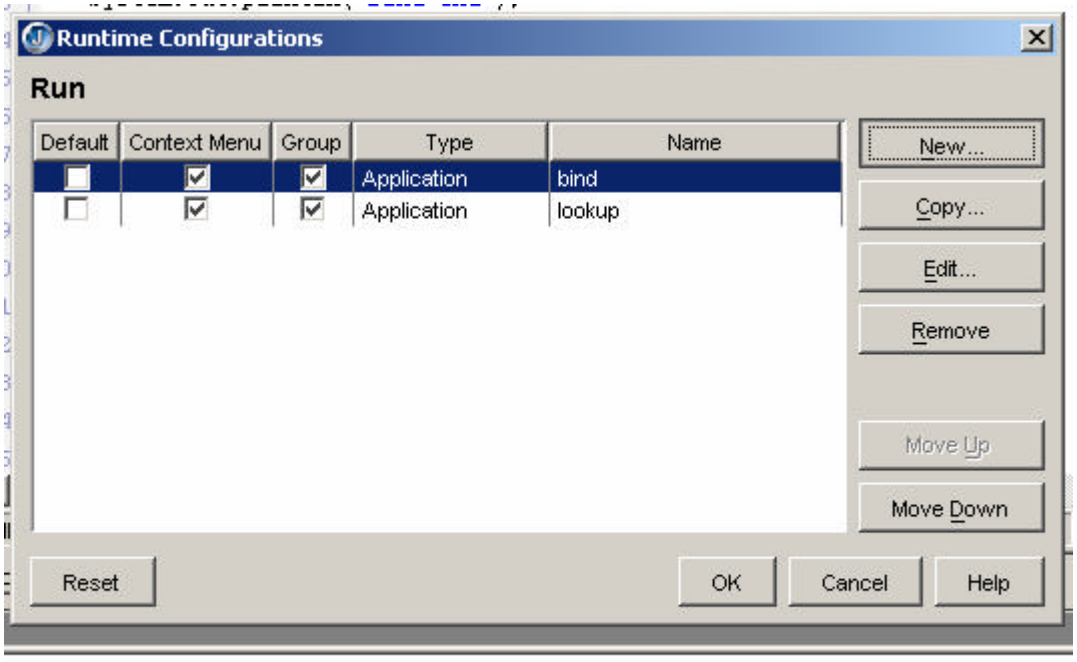


图 WSJ－8 设置运行时参数

在 Edit Runtime Configuration **窗口中做以下设置(建立运行项**"bind"**)：**

Name: bind

Build target: <None>

Type: Application

Application parameters: bind

如图 WSJ－9 **所示。**

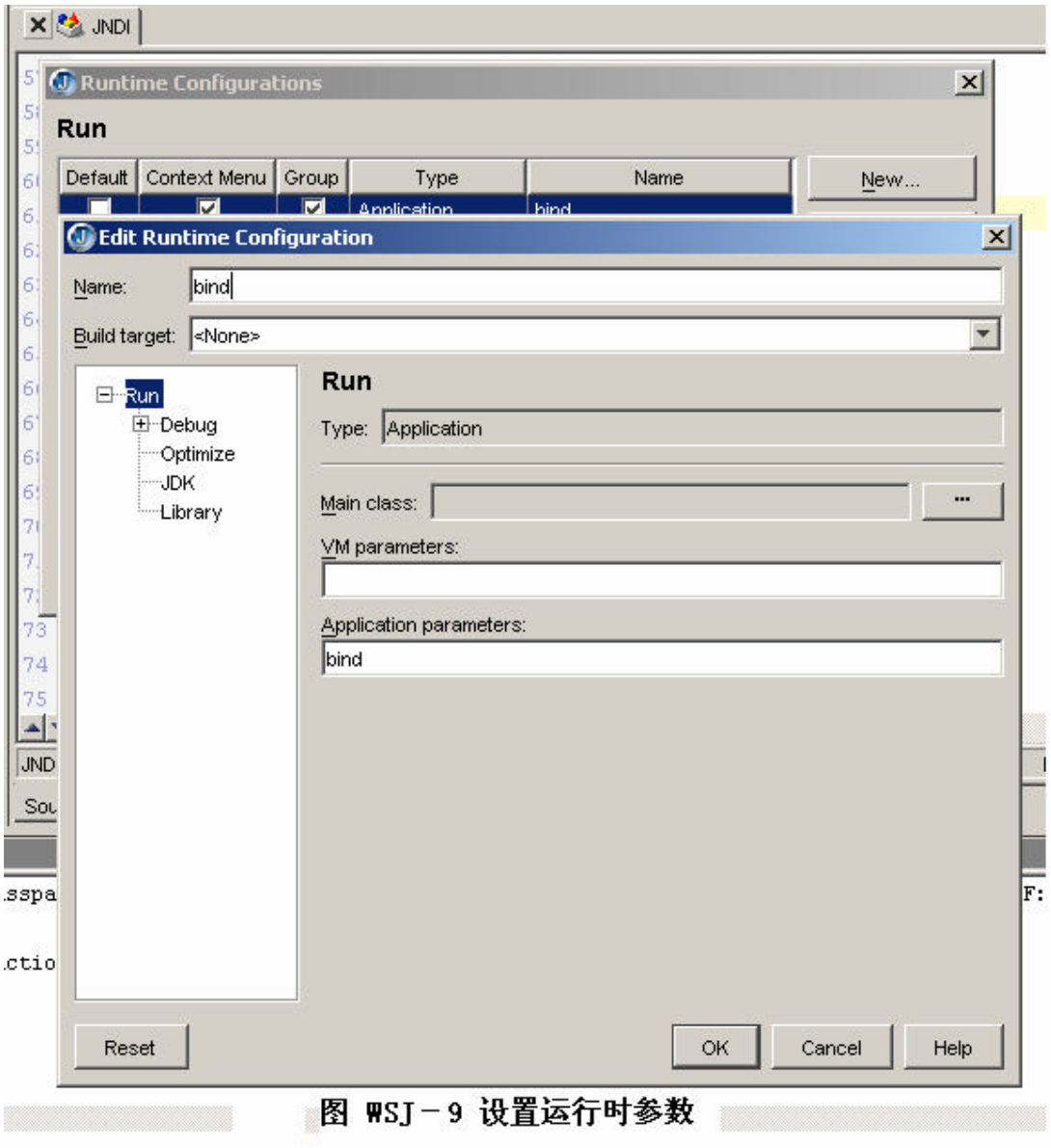


图 WSJ－9 设置运行时参数

以同样的方法建立运行项 lookup**，该运行项的属性设置如下：**

Name: lookup

Build target: <None>

Type:Application

Application parameters: lookup

5、**用参数运行类** JNDI

在运行改类前，要保证 WebLogic Server **已经正常启动，否则运行出错。用鼠标右键单击** JNDI.java **，在弹出的快捷菜单中选择** Run→Use “bind”**菜单项，此时将用参数** bind **运行改类，即实现绑定操作。如图** WSJ－10 **所示。**

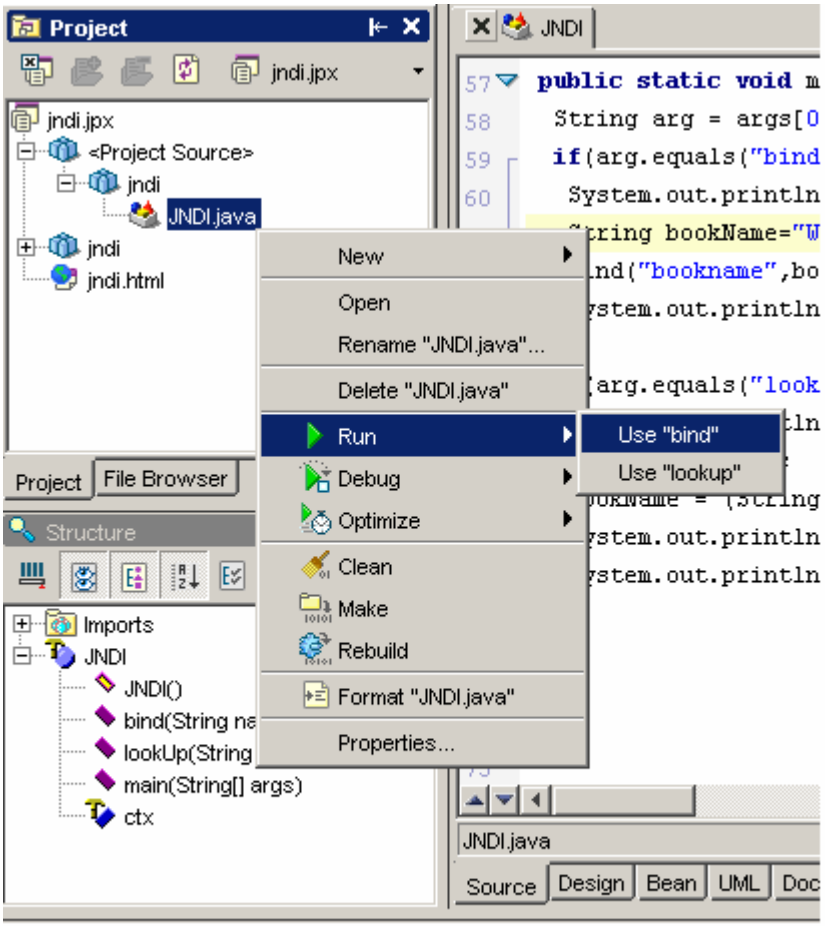


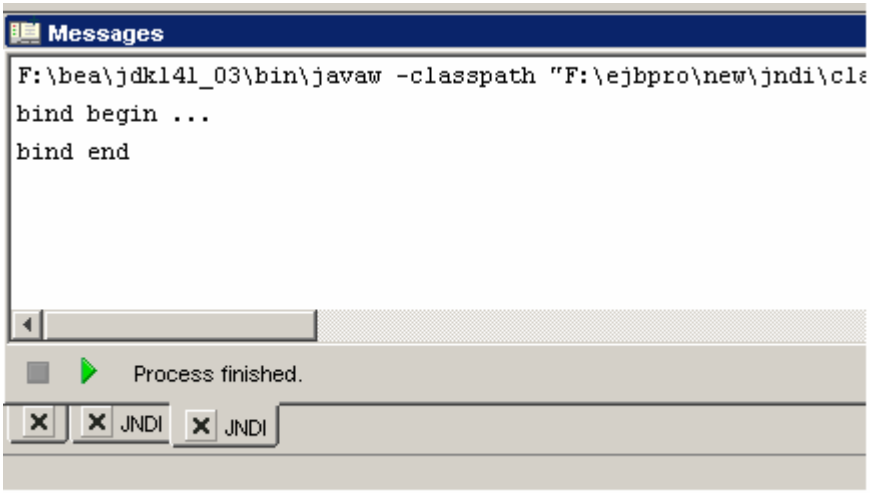
图 WSJ－10 以参数bind运行类JNDI

如果没有异常的话，在 JBuilderX **下面的** message **窗口中输出：**

bind begin…

bind end

如图 WSJ－11 **所示。**



WSJ－11 以参数bind正常运行改类

6、**启动** WebLogic Server **控制台并登录，如图** WSJ-2 **所示查看** JNDI Tree，**此时，**JNDI Tree **中新增了一个** bookname **结点。如图** WSJ－12 **所示。**

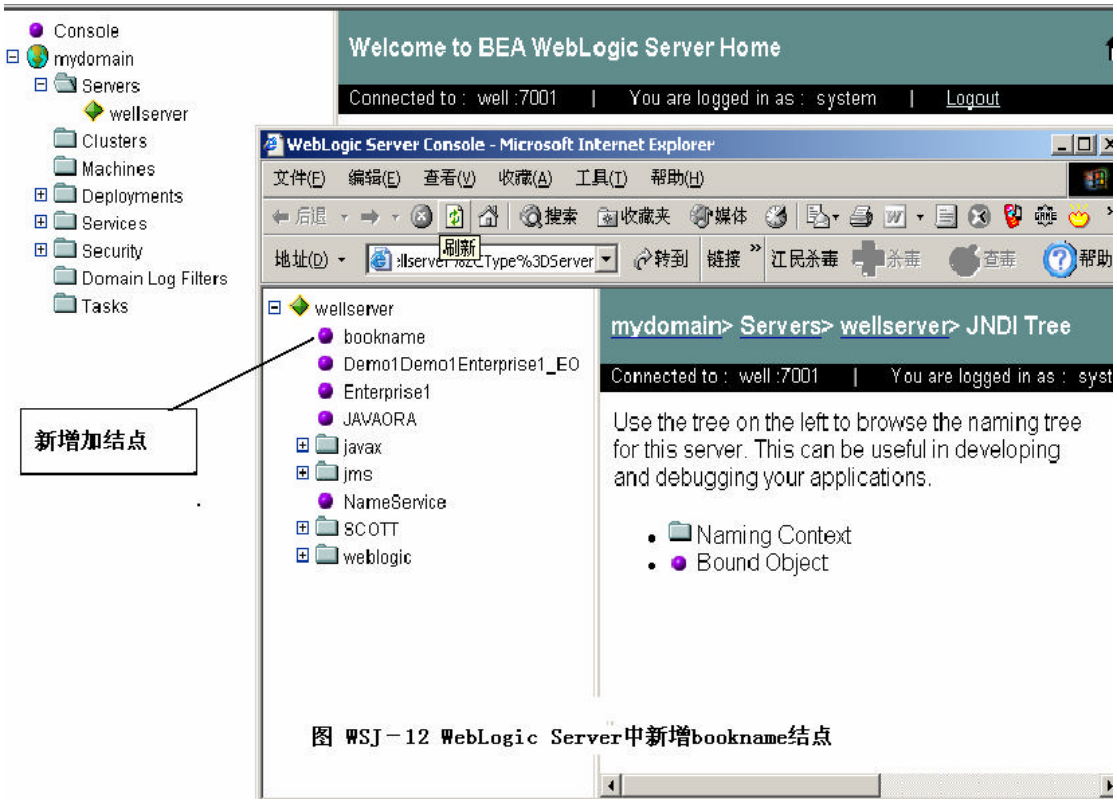


图 WSJ－12 WebLogic Server中新增bookname结点

结点 bookname **的详细信息如图** WSJ－13 **所示：**

图 WSJ－13 bookname详细信息

7、以 lookup 运行项运行改类，输出信息如图 WSJ－14 所示：

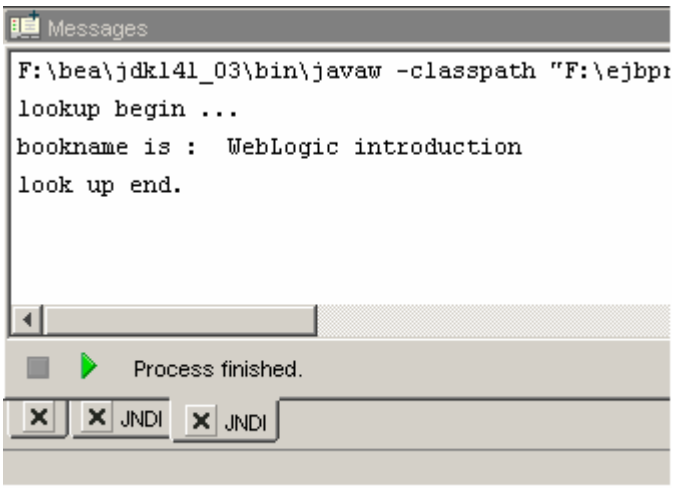


图 WSJ－14 lookup运行项结果

以上即是一个简单但完整的 WebLogic Server **下的** JNDI **的例子。除了例子中用到的，命名服务中上下文操作还包括：**

绑定操作 bind(), rebind()

解除绑定操作 unbind()

查询操作 lookup()

列举操作 list(), listBindings()

创建/删除子上下文操作 createSubContext(),destorySubcontext()

具体可参见相关文档。

E_mail:lhbing_7777@yahoo.com.cn