JNDI 原理 以及JBoss Demo - hekeji - JavaEye技术网站（转载）

汲取者 收录于2010-04-22 阅读数： 公众公开 原文来源

摘要:

本文详细介绍了JNDI的架构与实现,JNDI的工作原理，并给出了具体代码，帮助读者更理解J2EE主要常用技术---JNDI.本文为系列文章的第一 篇，其它相关文章会在近期推出。

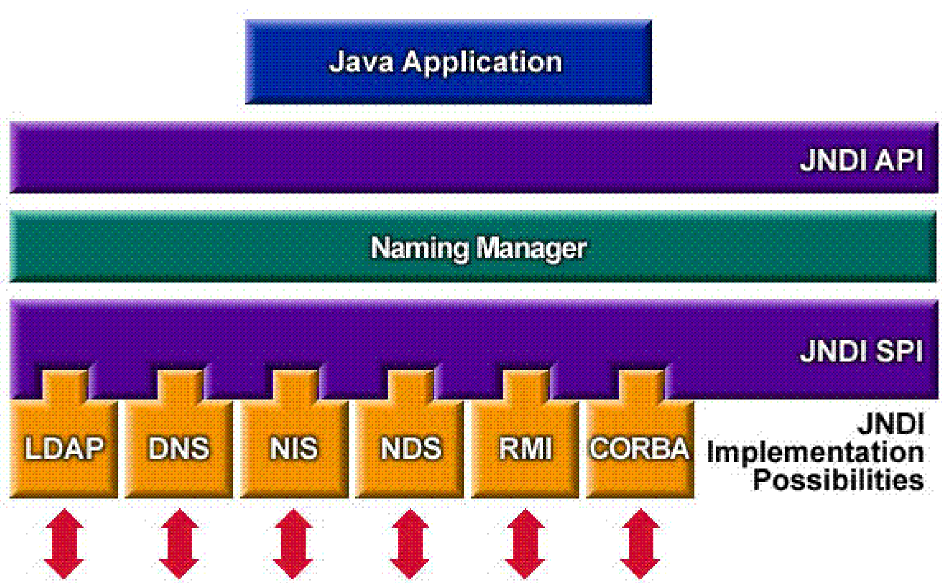
名词解释

jndi是Java 命名和目录接口（Java Naming and Directory Interface，JNDI）的简称.从一开始就一直是 Java 2 平台企业版（JEE）的核心技术之一。在JMS，JMail,JDBC,EJB等技术中，就大量应用的这种技术。

为什么会有jndi

jndi诞生的理由似乎很简单。随着分布式应用的发展，远程访问对象访问成为常用的方法。虽然说通过 Socket等编程手段仍然可实现远程通信，但按照模式的理论来说，仍是有其局限性的。RMI技术，RMI-IIOP技术的产生，使远程对象的查找成为了 技术焦点。JNDI技术就应运而生。JNDI技术产生后，就可方便的查找远程或是本地对象。

JNDI的架构与实现



JNDI的架构与JDBC的架构非常类似.JNDI架构提供了一组标准命名系统的API,这些API在JDK1.3之前是作为一个单独的扩展包 jndi.jar(通过这个地址下载),这个基础API构建在与SPI之上。这个API提供如下五个包

* javax.naming
* javax.naming.directory
* javax.naming.event

* javax.naming.ldap
* javax.naming.spi

在应用程序中,我们实际上只使到用以上几个包的中类.具体调用类及通信过程对用户来说是透明的.
JNDI API提供了访问不同JNDI服务的一个标准的统一的实现,其具体实现可由不同的 Service Provider来完成。前面讲的为第一层JNDI API层.
最下层为JNDI SPI API及其具体实现。

图中所列的一些SPI可从http://java.sun.com/products/jndi/downloads/index.html下 载.

attachments/200810/1170337664.gif
它包括了几个增强和下面的命名/目录服务提供者:

* LDAP(Lightweight Directory Access Protocol)服务提供者
* CORBA COS（Common Object Request Broker Architecture Common Object Services）命名服务提供者
* RMI(Java Remote Method Invocation)注册服务提供者
* DNS(Domain Name System)服务提供者.
* FSSP(File System Service Provider)文件系统服务提供者
* 其它服务提供者

中间层为命名管理层。其功能应该由JNDI SPI来完成。上层为JNDI API,这个API包在Java 2 SDK 1.3及以上的版本中已经包括。
前面讲解的只是作为应用程序客户端的架构实现,其服务端是由SPI对应的公司/厂商来实现的,我们只需将服务端的相关参数传给JNDI API就可以了,具体调用过程由SPI来完成.
JNDI工作原理

下面通过一个示例程序来说明JNDI工作原理(代码为自解释).
Java 代码

```
1.  /*
2.  * Created on 2005-3-4
3.  *
4.  * To change the template for this generated file go to
5.  * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
6.  */
7.  package com.sily.jndi;
8.
9.  import java.io.FileInputStream;
10. import java.util.Properties;
11.
12. import javax.naming.Context;
13. import javax.naming.InitialContext;
14.
15. /**
16. * @author shizy
17. *
18. * To change the template for this generated type comment go to
19. * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
20. */
```

```
21. public class TestJbossJNDI {
22. /**
23. *
24. */
25. public TestJbossJNDI() {
26. super();
27. // TODO Auto-generated constructor stub
28. }
29. public static void main(String[] args) { try {
30. Properties env = new Properties();
31. //载入jboss的SPI相关参数,包括初始上下文工厂，服务URL，等等
32. env.load(new FileInputStream("jbossJndi.properties"));
33. env.list(System.out);
34. //通过JNDI api 初始化上下文
35. InitialContext ctx = new javax.naming.InitialContext(env);
36. System.out.println("Got context");
37. //create a subContext
38. ctx.createSubcontext("/sylilzy");
39. ctx.createSubcontext("sylilzy/sily");
40. //rebind a object
41. ctx.rebind("sylilzy/sily/a", "I am sily a!");
42. ctx.rebind("sylilzy/sily/b", "I am sily b!");
43.
44. //lookup context
45. Context ctx1=(Context)ctx.lookup("sylilzy");
46. Context ctx2=(Context)ctx1.lookup("/sylilzy/sily");
47. ctx2.bind("/sylilzy/g", "this is g");
48. //lookup binded object
49. Object o;
50. o=ctx1.lookup("sily/a");
51. System.out.println("get object from jndi:"+"get object from jndi:"+o);
52. //rename the object
53. ctx2.rename("/sylilzy/g", "g1");
54. o=ctx2.lookup("g1");
55. System.out.println("get object from jndi:"+"get object from jndi:"+o);
56.
57. } catch (Exception e) {
58. e.printStackTrace();
59. }
60. }
61. }
62. 结 果输出如下:
63.
64. -- listing properties --
65. java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
66. java.naming.provider.url=jnp://localhost:1099
67. java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
68. Got context
69. get object from jndi:I am sily a!
70. get object from jndi:this is g
71.
```

程序中jbossJndi.properties文件的内容为:
Java 代码

```
java.naming.factory.initial=org.jnp.interfaces.NamingContextFactory
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
java.naming.provider.url=jnp://localhost:1099
```

注意:要正确运行示例程序,请启动jboss,并将jboss的jbossall-client.jar文件放入classpath中。

上述示例程序在jboss服务器的jndi树上建立了几个上下文,并bind了几对象,大家可通过附录中的代码或其它工具查看
查看结果为:
Java 代码

```
------------------------------
/sylilzy/sily
------------------------------
/sylilzy/sily/b:I am sily b!
/sylilzy/sily/a:I am sily a!
/sylilzy/sily/g1:this is g
------------------------------
------------------------------
```

上述程序中，我们的代码只涉及到了jndi API，其它细节如初始化jboss jndi的初始上下文，建立网络连接，与服务器通信，对我们来说都是透明的，另外，我们将jboss jndi的spi包中的类名作为参数传入了程序中,要访问一个远程对象，我们所做的就这么多。

下面，再提供一个例子，与上例不同，我们不需要 jboss,我们使用sun的FSSP(File System Service Provider)文件系统服务提供者.注意在这个例子中要使用到前面所说的File System Service Provider for the java Naming and Directory InterfaceTM (JNDI)相关类（下载）。
Java 代码

```
/*
 * Created on 2005-3-1
 *
 * To change the template for this generated file go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
package com.sily.jndi;
import java.io.FileInputStream;
import java.util.Properties;
import javax.naming.*;
import javax.naming.Context;
import javax.naming.InitialContext;
/**
 * @author shizy
 *
```

```
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class JndiTest1 {
    /**
     *
     */
    public JndiTest1() {
        super();
        // TODO Auto-generated constructor stub
    }
    public static void main(String[] args) {
        try {
            Properties env = new Properties();
            env.load(new FileInputStream("fileSystemService.properties"));
            env.put(Context.PROVIDER_URL, "file:///c:/");
            Context ctx = new InitialContext(env);
            ctx.createSubcontext("sylilzy");

            NamingEnumeration list = ctx.list("/");
            while (list.hasMore()) {
                NameClassPair nc = (NameClassPair) list.next();
                System.out.println(nc);
            }

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

上例中fileSystemService.properties文件的内容为：java.naming.factory.initial=com.sun.jndi.fscontext.RefFSContextFactory

这个例子较简单，运行后，它会列出C:\下所有的文件和目录，另外你会发现有一个新目录被创建了.本例不同于上例，它并不需要服务端，因为它访问 的是文件系统.有关帮助可查阅包内的相关文档。

通过对比这两个例子，应该JNDI的工作原理有了一个大致的了解。

总结：

jndi技术体现了分布式应用的优点，同进它的产生也为分布式对象提供了统一的访问接口。由于篇幅所限，对目录的操作本文未作介绍，其它内容将在 接下来的系列中讨论。要对JNDI技术作全面的了解，请参阅参考资料.要对于JNDI技术深入学习，仍有许多地方值得进一步了解，例如EJB容器所使用的 JNDI所提供的对象就有 Local和Remote之分，对于Local Object,对于不同的JVM是不可访问的；对于远程对象的访问，还涉及到Java安全机制。

附录：

查看jboss jndi内容的代码:

Java 代码

```
1.  //----------------------------------------
2.
3.  /*
4.   * Created on 2005-3-4
5.   *
6.   * To change the template for this generated file go to
7.   * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
8.   */
9.  package com.sily.jndi;
10. import java.io.FileInputStream;
11. import java.util.Properties;
12.
13. import javax.naming.*;
14. import javax.naming.Context;
15. import javax.naming.InitialContext;
16. /**
17.  * @author shizy
18.  *
19.  * To change the template for this generated type comment go to
20.  * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
21.  */
22. public class ListJbossJndi {
23.     /**
24.      *
25.      */
26.     public ListJbossJndi() {
27.         super();
28.         // TODO Auto-generated constructor stub
29.     }
30.     public static void main(String[] args) {
31.         try {
32.             Properties env = new Properties();
33.             env.load(new FileInputStream("jbossJndi.properties"));
34.             //env.list(System.out);
35.             Context ctx = new InitialContext(env);
36.             listCtx(ctx.lookup("sylilzy"));
37.         }
38.         catch (Exception e) {
39.             e.printStackTrace();
40.         }
41.     }
42.     static void listCtx(Object o){
43.         if(!(o instanceof Context))log(":"+o);
44.         else {
45.             log("\n------------------------------");
46.         try {
47.             Context ctx=(Context)o;
48.             //log(ctx.getNameInNamespace()+"/:");
49.             NamingEnumeration list=ctx.listBindings("");
```

```
            while(list.hasMore()){
                Binding bind=(Binding)list.next();
                log("\n/"+ctx.getNameInNamespace()+"/"+bind.getName());
                listCtx(bind.getObject());
            }
            log("\n----------------------------");
        }
        catch (NamingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    }
    static void log(Object o){
        System.out.print(o);
    }
}
```

上一篇：JNDI配置原理详解 - fengzl - JavaEye技术网站
下一篇：Axis2体系结构中文手册

(汲取者 的分类目录 [我的图书馆])

相关文章

- 开发基于JNDI的应用程序 2006-08-30 鬼迷心窍
- JNDI配置原理详解(转)- 高山流水 - 新浪BLOG 2007-07-12 byond
- BEA WebLogic Server8.1 JMS入门 2006-08-24 lwj888
- Tomcat 数据库连接池的相关参数 2006-08-08 鬼迷心窍
- EJB3.0开发指南之有状态会话Bean(转) - JRen大鹏... 2008-01-30 sealyu
- 使用JNDI操作LDAP（1） 2009-03-08 种子张
- JSP中tomcat的SQL Server2000数据库连接池的... 2005-09-24 duduwolf
- JNDI全攻略之(二) 2007-09-21 coolper

查看更多文章>>

发表评论

发送评论时内容自动复制到剪切板 发送