JNDI

- 1 JNDI概述
- 2 使用JNDI连接数据库
  - 2.1 配置数据源
  - 2.2 在程序中引用数据源
- 3 JNDI的使用
  - 3.1 创建初始上下文环境
  - 3.2 获取JNDI对象

1 JNDI概述

JNDI 是 Java 命名与目录接口（Java Naming and Directory Interface）的缩写，是 Java EE 架构中重要的规范之一，为开发人员提供了查找和访问各种命名和目录服务的通用、统一的接口。

2 使用JNDI连接数据库

使用 JNDI 连接数据库的方法：首先，在 JavaEE 容器中配置 JNDI 参数，定义一个数据源，也就是 JDBC 引用参数，给这个数据源设置一个名称；然后，在程序中，通过数据源名称引用数据源从而访问后台数据库。

下面以 JBoss AS7 为例，说明具体操作过程。

2.1 配置数据源

1. 为 JBoss 添加 MySQL 数据驱动

首先，在 JBOSS_HOME\modules\com 路径下新建文件夹 mysql，在 mysql 文件夹下再新建文件夹 main，并将 MySQL 驱动 jar 包（这里是 mysql-connector-java-5.1.26-bin.jar）复制到 main 文件夹下。同时在 main 目录中新建 module.xml 文件，文件内容如下：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- 注意：name 属性值与文件夹名保持一致 -->
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
    <resources>
        <resource-root path="mysql-connector-java-5.1.26-bin.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
        <module name="javax.servlet.api" optional="true"/>
    </dependencies>
</module>
```

接下来，修改 standalone.xml 配置文件。在 JBOSS_HOME\standalone\configuration 路径下找到 standalone.xml 文件，打开文件后找到`<datasources>...<drivers>...</drivers>...</datasources>`标签，在`drivers`标签中添加如下的配置：

```xml
<!-- 注意：module 属性值与 module.xml 文件中的 name 属性值保持一致 -->
<driver name="mysql" module="com.mysql">
    <driver-class>
        com.mysql.jdbc.Driver
    </driver-class>
    <xa-datasource-class>
        com.mysql.jdbc.jdbc2.optional.MysqlXADataSource
    </xa-datasource-class>
</driver>
```

2. 为 JBoss 添加 MySQL 数据源

打开上述的 standalone.xml 文件，找到`<datasources>`标签，里面有一个名为 ExampleDS 的默认数据源。在`<datasources>`标签中、与 ExampleDS 数据源同级的位置添加自定义的 MySQL 数据源：

```xml
<datasource jndi-name="java:jboss/datasources/MySqlDS" pool-name="MySqlDS"
            enabled="true" use-java-context="true">
    <connection-url>
        jdbc:mysql://localhost:3306/database-name
    </connection-url>
    <driver>mysql</driver> <!-- 与上面配置的驱动名称保持一致 -->
    <security>
        <user-name>root</user-name>
        <password>123456</password>
    </security>
</datasource>
```

还可以添加一个支持分布式数据库的`xa-datasource`：

```
<xa-datasource jndi-name="java:jboss/datasources/MySqlXADS" pool-name="MySqlXADS"
               enabled="true" use-java-context="true">
    <xa-datasource-property name="URL">
        jdbc:mysql://localhost:3306/xa-database-name
    </xa-datasource-property>
    <driver>mysql</driver>
    <security>
        <user-name>root</user-name>
        <password>123456</password>
    </security>
</xa-datasource>
```

2.2 在程序中引用数据源

```
Connection conn = null;
try
{
    Context ctx = new InitialContext(); // 获取上下文环境
    Object datasourceRef = ctx.lookup("java:jboss/datasources/MySqlDS"); // 引用数据源
    DataSource ds = (Datasource) datasourceRef;
    conn = ds.getConnection();
    /* 使用 conn 进行数据库 SQL 操作 */
    // ......
    c.close();
}
catch (Exception e)
{
    e.printStackTrace();
}
finally
{
    if(conn != null)
    {
        try
        {
            conn.close();
        }
        catch (SQLException e) { }
    }
}
```

从上面的代码可以看出，通过 JNDI 引用数据源的编程代码量与直接使用 JDBC 相差无几，但是现在的程序可以不用关心具体 JDBC 参数了。在系统部署后，如果数据库的相关参数变更，只需要重新配置 standalone.xml 修改其中的`datasource`数据源中的参数即可，只要保证数据源的名称不变，那么程序源代码就无需修改。

由此可见，JNDI 避免了程序与数据库之间的紧耦合，使应用更加易于配置、易于部署。

3 JNDI的使用

3.1 创建初始上下文环境

不管进行什么操作，都要先创建初始上下文。

如果是客户端使用服务端上 JNDI 对象，创建初始上下文环境时，必须指定上下文工厂、服务器的命名和目录管理地址以及用户名和密码。

在 JBoss AS7 中，应用程序使用 JNDI 时，为了保证安全性，必须要进行用户验证。因此，要先在 JBoss AS7 中创建一个应用程序用户。在 JBoss_HOME\bin 目录下找到 add-user.bat 文件，双击执行，弹出如下命令行窗口。

```
What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a):
```

输入 b，按下回车，效果如下所示。

```
What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a): b

Enter the details of the new user to add.
Realm (ApplicationRealm) :
```

在`Realm (ApplicationRealm) :`处按回车，选用默认配置，效果如下所示。

```
Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username :
```

输入用户名（这里以 testJNDI 为例）并按下回车，效果如下所示。

```
Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : testJNDI
Password :
```

输入密码（密码不会显示出来），按下回车，效果如下所示。

```
Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : testJNDI
Password :
Re-enter Password :
```

重新输入密码，按下回车，效果如下所示。

```
Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : testJNDI
Password :
Re-enter Password :
What roles do you want this user to belong to? (Please enter a comma separated list, or
leave blank for none) :
```

输入 testrole，按下回车，效果如下所示。

```
Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : testJNDI
Password :
Re-enter Password :
What roles do you want this user to belong to? (Please enter a comma separated list, or
leave blank for none) : testrole
About to add user 'testJNDI' for realm 'ApplicationRealm'
Is this correct yes/no?
```

输入 yes，按下回车，效果如下所示。

```
Enter the details of the new user to add.
Realm (ApplicationRealm) :
Username : testJNDI
Password :
Re-enter Password :
What roles do you want this user to belong to? (Please enter a comma separated list, or
leave blank for none) : testrole
About to add user 'testJNDI' for realm 'ApplicationRealm'
Is this correct yes/no? yes
Added user 'testJNDI' to file 'D:\JBoss\jboss-as-7.1.1.Final\standalone\configuration\
application-users.properties'
Added user 'testJNDI' to file 'D:\JBoss\jboss-as-7.1.1.Final\domain\configuration\
application-users.properties'
Added user 'testJNDI' with roles testrole to file 'D:\JBoss\jboss-as-7.1.1.Final\
standalone\configuration\application-roles.properties'
Added user 'testJNDI' with roles testrole to file 'D:\JBoss\jboss-as-7.1.1.Final\domain\
configuration\application-roles.properties'
请按任意键继续. . .
```

至此，JBoss AS7 的应用程序用户创建完成。

在客户端上创建上下文的例子如下所示：

```
Properties prop = new Properties();
// 服务器的命名和目录管理地址
prop.put(Context.PROVIDER_URL, "remote://localhost:4447");
// 初始化上下文环境工厂
// 需要引入 jboss-client.jar 包，位于 JBOSS_HOME\bin\client 目录下
prop.put(Context.INITIAL_CONTEXT_FACTORY,
         org.jboss.naming.remote.client.InitialContextFactory.class.getName());
// 用户验证
prop.put(Context.SECURITY_PRINCIPAL,
         System.getProperty("username", "testJND"));
prop.put(Context.SECURITY_CREDENTIALS,
         System.getProperty("password", "123456"));
// 初始化上下文
Context context = new InitialContext(prop);
```

可以看出在调用`InitialContext()`方法中需要传递参数`Properties`，它包含 4 个值对，分别为服务器的命名和目录管理地址`PROVIDER_URL`、初始化上下文环境工厂`INITIAL_CONTEXT_FACTORY`、用户名`SECURITY_PRINCIPAL`和密码`SECURITY_CREDENTIALS`。

在服务器端创建上下文的例子如下所示：

```
Context ctx = new InitialContext();
```

在调用`InitialContext()`方法中不需要传递参数。这是因为我们是在服务器端调用`InitialContext()`方法，而`InitialContext()`方法中需要的参数服务器都可以从配置文件中找到，无需进行远程传递。

3.2 获取JNDI对象

通常远程客户端使用`lookup()`方法可以获取 JNDI 对象，代码如下所示：

```
// 实际访问的是 java:jboss/exported/jms/topic/test
Topic test = (Topic) ctx.lookup("jms/topic/test");
```