# 数据源

## 一、DataSource

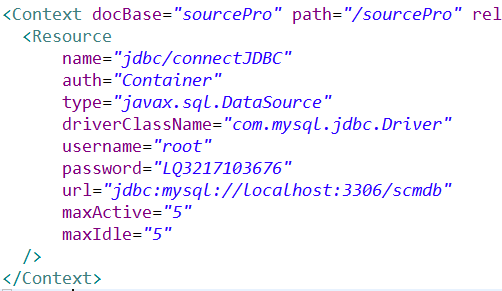
一般在操作数据库时，若不断的创建Connection对象用来连接数据库和关闭数据库，这样会耗费很多资源。所以一般通过连接池(Connection Pool)来管理Connection对象，连接池中存放着一些可重复利用的Connection对象。当需要连接数据库时，通过数据源(DataSource)从连接池中获取Connection对象，当使用完这个连接对象之后，继续放回连接池中，这样连接对象就可以重复使用。

DataSource被JNDI绑定。JNDI(java naming and directory interface)资源目录是用来进行资源定位的，它是给当前应用服务器所管理的所有资源一个唯一的标识，包括数据库，网页，文件，连接池等等。JNDI也是存放对象信息的一种机制或协议。

### 1.在服务器上对DataSource进行配置

#### ⑴步骤一：

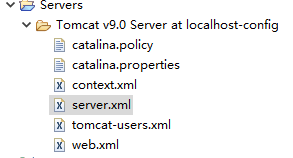
**方法一**：**在server.xml中配置如下信息**。这种方式会破坏server.xml，若一台机器上使用多个tomcat服务器就会产生麻烦。在server.xml中配置的是全局的JNDI。



注意，若将项目移除了，则上面配置的这个信息也会被移除，要重新配置。

**具体过程：**

**①**直接在Servers中的server.xml中进行配置，这个前提Server Locations中使用的是安装的tomcat，而不是workspace中的metadata。

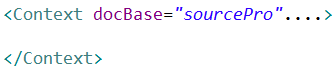


**②**打开server.xml文件，在文件中找到**<Context..../>**标签。注意只有项目发布到tomcat中，在server.xml文件中才有如下的标签。其中docBase的内容是项目的存放路径，可以是绝对路径和相对路径，若是相对路径如docBase=”sourcePro”则这个项目必须在tomcat的webApps目录下。

<Context docBase=*"sourcePro"* path=*"/sourcePro".....*/></Host>

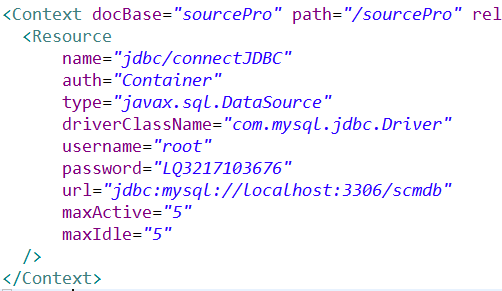
**③**将**<Context.../>单节点标签该为双节点标签**。

：单节点

：双节点

**④**然后在这个双节点标签中补<Resource/>标签中的内容。注意<Resource/>标签是单节点的。

其中auth=*"Container"*和type=*"javax.sql.DataSource"***是不能变的***。* driverClassName=*"com.mysql.jdbc.Driver"*表示连接的驱动*。*



**name**：**给这个资源配置的名称**。符合java规范即可。

**type**：**代表当前资源的类型**。type=“javax.sql.DataSource”表示当前资源是数据源。

**auth**：**表示当前资源的管理者**。auth=“Container”表示当前资源的管理者是容器。

**maxActive**：表当前活跃的最大数量。即可以同时有多少个Connection被使用。

**maxIdle**：表当前最大的空闲数量，即同时可以有多少个空闲。

**maxWait**：最大的等待时间，单位是毫秒。

**方法二**：在web项目中的webapp目录下的META-INF目录 (若没这个则新建一个)下，创建一个context.xml文件，然后将上面<Resource>中的内容复制到这个文件中。(**推荐使用这种方式**)。这种方式配置的是局部的JNDI。

#### **⑵步骤二：**

在web工程的web.xml中配置如下信息：

注意这是在Web工程中的WEB-INFO目录下的web.xml文件，而不是Servers文件夹下的web.xml文件。

下面的name、type、auth的内容和server.xml中配置的一样。



#### ⑶步骤三：

**把连接数据库的jar包拷贝导tomcat的lib目录下**。特别注意不是直接将mysql-connector-java-5.1.15-bin.jar包放到web项目中的WEB-INFO目录下，而是放到tomcat的lib目录下。

#### ⑷步骤四：

将原来DbUtil包中获取连接对象的代码改成下面这种方式，Dao包和Service包中的内容还是和原来的一样。

注意下面这部分代码不能直接写在main方法中运行，因为是由tomcat来管理资源的，所以下面的代码应该和tomcat关联起来。



上面代码的实际运行过程：首先程序代码获取初始化的JNDI 环境，调用Context.lookup()方法从JNDI服务提供者那里获一个DataSource对象。中间层JNDI 服务提供者返回一个DataSource对象给当前的Java 应用程序，这个DataSource 对象代表了中间层服务上现存的缓冲数据源。

应用程序调用DataSource 对象的getConnection()方法，当该方法被调用时，首先查看连接池中有没有现有的Connection实例对象，若有将该实例对象直接给应用程序，若没有则产生一个新的Connection实例给应用程序使用。

当应用程序不需要使用Connection 对象时，可以调用Connection 接口的close()方法。注意这种情况下close()方法并没有关闭事实上的数据库连接，仅仅是释放了被应用程序占用的数据库连接，并将它还给数据库连接缓冲池，数据库连接缓冲池会自动将这个数据库连接交给请求队列中下一个的应用程序使用。