1. 将数据库驱动程序的 JAR 文件放在 Tomcat 的 common/lib 中；

2. 在 server.xml 中设置数据源，以 MySQL 数据库为例，如下：
在<GlobalNamingResources> </GlobalNamingResources>节点中加入，

```
<Resource
name="jdbc/DBPool"
type="javax.sql.DataSource"
password="root"
driverClassName="com.mysql.jdbc.Driver"
maxIdle="2"
maxWait="5000"
username="root"
url="jdbc:mysql://127.0.0.1:3306/test"
maxActive="4"/>
```

属性说明：name，数据源名称，通常取”jdbc/XXX”的格式；
type，”javax.sql.DataSource”；
password，数据库用户密码；
driveClassName，数据库驱动；
maxIdle，最大空闲数，数据库连接的最大空闲时间。超过空闲时间，数据库连接将被标记为不可用，然后被释放。设为 0 表示无限制。
MaxActive，连接池的最大数据库连接数。设为 0 表示无限制。
maxWait ，最大建立连接等待时间。如果超过此时间将接到异常。设为-1 表示无限制。

3. 在你的 web 应用程序的 web.xml 中设置数据源参考，如下：
在<web-app></web-app>节点中加入，

```
<resource-ref>
  <description>MySQL DB Connection Pool</description>
  <res-ref-name>jdbc/DBPool</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

子节点说明： description，描述信息；
res-ref-name，参考数据源名字，同上一步的属性name；
res-type，资源类型，”javax.sql.DataSource”；
res-auth，”Container”；
res-sharing-scope，”Shareable”；

4. 在web应用程序的context.xml中设置数据源链接，如下：
在<Context></Context>节点中加入，
<ResourceLink
name="jdbc/DBPool"
type="javax.sql.DataSource"
global="jdbc/DBPool"/>
属性说明：name，同第2步和第3步的属性name值，和子节点res-ref-name值；
type，同样取”javax.sql.DataSource”；
global，同name值。

至此，设置完成，下面是如何使用数据库连接池。
1. 建立一个连接池类，DBPool.java，用来创建连接池，代码如下：

```
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class DBPool {
    private static DataSource pool;
    static {
         Context env = null;
          try {
              env = (Context) new InitialContext().lookup("java:comp/env");
              pool = (DataSource)env.lookup("jdbc/DBPool");
              if(pool==null)
                  System.err.println("'DBPool' is an unknown DataSource");
```

```
                } catch(NamingException ne) {
                    ne.printStackTrace();
            }
        }
    public static DataSource getPool() {
        return pool;
    }
}
```

设置基本上主要有两种方法我们以 MySQL+TOMCAT 为例
1.把 DataSource 设置到我们的 WEB 项目中，下面详细的介绍下：
第一步：在我们的 WEB 项目中的 META-INF 文件夹下建立一个 context.xml

Xml 代码

```xml
<?xml version='1.0' encoding='utf-8'?>

<Context>

    <Resource name="jdbc/mysql"
       auth="Container"
       type="javax.sql.DataSource"
       driverClassName="com.mysql.jdbc.Driver"
       url="jdbc:mysql://localhost/bbs"
            username="root"
            password="root"
            maxActive="50"
            maxIdle="20"
            maxWait="10000" />

     </Context>
```

第二步：在我们的 WEB 项目下的 WEB-INF 文件夹下建立一个 web.xml(如果存在了就不用了，直接修改就行了)
(这几天测试了一下，不做这步也可以，0(∩_∩)0 哈哈~省事了)

Xml 代码

```xml
<resource-ref>
    <description>DB Connection</description>
    <res-ref-name>jdbc/mysql</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
</resource-ref>
```

第三步：我们就可以用代码来获取 Connection 对象了

Java 代码

```
1. package xushun.util;
2.
3. import java.sql.*;
4. import javax.sql.*;
5. import javax.naming.*;
6.
7. public class DBHelper {
8.
9.     public static Connection getConnection() throws SQ
   LException,NamingException
10.         {
11.             // 初始化查找命名空间
12.             Context initContext = new InitialContext()
   ;
13.             Context envContext = (Context)initContext.
   lookup("java:/comp/env");
14.             // 找到 DataSource
15.             DataSource ds = (DataSource)envContext.lo
   okup("jdbc/mysql");
16.             return ds.getConnection();
17.         }
18.     }
```

2.把 DataSource 设置到我们的 Tomcat 中，下面详细的介绍下（测试用的 JAVA 代码和上面的一样就不帖出了）：
这里我查到的设置方法就有了一点区别了。有的人把 DataSource 设置在 Tomcat 的 server.xml 文件的 GlobalNamingResources 下面，然后在 context.xml 中去映射。有的直接就写在 context.xml 中了
先说下在 server.xml 添加 DataSource
第一步：在 Tomcat 的 conf 中的 server.xml 文件中找到

Xml 代码

```
1. <GlobalNamingResources>
```

```
    <!--
     Editable user database that can also be used by
          UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
      </GlobalNamingResources>
```

修改为

Xml 代码

```
<GlobalNamingResources>
    <!--
     Editable user database that can also be used by
          UserDatabaseRealm to authenticate users
    -->
    <Resource name="UserDatabase" auth="Container"
              type="org.apache.catalina.UserDatabase"
              description="User database that can be updated and saved"
              factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
              pathname="conf/tomcat-users.xml" />
        <Resource name="jdbc/bbs"
                  auth="Container" type="javax.sql.DataSource"
                  driverClassName="com.mysql.jdbc.Driver"

                  maxIdle="20"
                  maxWait="5000"
                  username="root"
                  password="admin"
```

```
17.                 url="jdbc:mysql://localhost:3306/bbs"

18.                 maxActive="100"
19.                 removeAbandoned="true"
20.                 removeAbandonedTimeout="60"
21.                 logAbandoned="true"/>
22.     </GlobalNamingResources>
```

第二步：在 Tomcat 的 conf 文件夹下的 context.xml 中加入

Xml 代码

```
1. <ResourceLink name="jdbc/bbs" global="jdbc/bbs" type="
   javax.sql.DataSource"/>
```

第三步：就是在 WEB 项目的 WEB-INF 中的 web.xml 添加

Xml 代码

```
1. <resource-ref>
2.     <description>DB Connection</description>
3.     <res-ref-name>jdbc/mysql</res-ref-name>
4.     <res-type>javax.sql.DataSource</res-type>
5.     <res-auth>Container</res-auth>
6. </resource-ref>
```

还有就是在 Tomcat 文档中提到的方法，直接修改 context.xml 文件了

在 Tomcat 的 conf 文件夹下的 context.xml 中加入

Xml 代码

```
1. <Resource name="jdbc/bbs"
2.                 auth="Container" type="javax.sql.DataSou
   rce"
3.                 driverClassName="com.mysql.jdbc.Driver"

4.                 maxIdle="20"
```

```
5.                  maxWait="5000"
6.                  username="root"
7.                  password="admin"
8.                  url="jdbc:mysql://localhost:3306/bbs"

9.                  maxActive="100"
10.                      removeAbandoned="true"
11.                      removeAbandonedTimeout="60"
12.                      logAbandoned="true"/>
```

然后就是在 WEB 项目的 WEB-INF 中的 web.xml 添加

Xml 代码

```
1. <resource-ref>
2.     <description>DB Connection</description>
3.     <res-ref-name>jdbc/mysql</res-ref-name>
4.     <res-type>javax.sql.DataSource</res-type>
5.     <res-auth>Container</res-auth>
6. </resource-ref>
```

就是这些了，如果有什么不太清楚的就留言，一起研究下。等以后我在搜集下资料整理出上面用到的 XML 文件中各个标签的属性及其代表的意思。有兴趣的也可以自己先查下。:-)

<td>JNDI 查找名称</td>　　<td>关联的引用</td>

<td>java:comp/env</td>　　<td>应用程序环境条目</td>

<td>java:comp/env/jdbc</td> <td>JDBC 数据源资源管理器连接工厂</td>

<td>java:comp/env/ejb</td>　<td>EJB 引用</td>

<td>java:comp/UserTransaction</td><td>UserTransaction 引用</td>

<td>java:comp/env/mail</td> <td>JavaMail 会话连接工厂</td>

<td>java:comp/env/url</td> <td>URL 连接工厂</td>

<td>java:comp/env/jms</td> <td>JMS 连接工厂和目标</td>

<td>java:comp/ORB</td> <td>应用程序组件之间共享的 ORB 实例</td>

数据库连接是一种关键的有限的昂贵的资源，这一点在多用户的网页应用程序中体现得尤为突出。对数据库连接的管理能显著影响到整个应用程序的伸缩性和健壮性，影响到程序的性能指标。数据库连接池正是针对这个问题提出来的。

数据库连接池负责分配、管理和释放数据库连接，它允许应用程序重复使用一个现有的数据库连接，而再不是重新建立一个;释放空闲时间超过最大空闲时间的数据库连接来避免因为没有释放数据库连接而引起的数据库连接遗漏。这项技术能明显提高对数据库操作的性能。

数据库连接池在初始化时将创建一定数量的数据库连接放到连接池中，这些数据库连接的数量是由最小数据库连接数来设定的。无论这些数据库连接是否被使用，连接池都将一直保证至少拥有这么多的连接数量。连接池的最大数据库连接数量限定了这个连接池能占有的最大连接数，当应用程序向连接池请求的连接数超过最大连接数量时，这些请求将被加入到等待队列中。数据库连接池的最小连接数和最大连接数的设置要考虑到下列几个因素：

1）最小连接数是连接池一直保持的数据库连接，所以如果应用程序对数据库连接的使用量不大，将会有大量的数据库连接资源被浪费;

2）最大连接数是连接池能申请的最大连接数，如果数据库连接请求超过此数，后面的数据库连接请求将被加入到等待队列中，这会影响之后的数据库操作。

3）如果最小连接数与最大连接数相差太大，那么最先的连接请求将会获利，之后超过最小连接数量的连接请求等价于建立一个新的数据库连接。不过，这些大于最小连接数的数据库连接在使用完

不会马上被释放，它将被放到连接池中等待重复使用或是空闲超时后被释放。

Tomcat6.0 连接池配置

1.配置 tomcat 下的 conf 下的 context.xml 文件,在之间添加连接池配置:

<Resource name="jdbc/sample"

auth="Container"

type="javax.sql.DataSource"

driverClassName="com.microsoft.sqlserver.jdbc.SqlserverDriver"

url="jdbc:sqlserver://localhost:8080;databaseName=test"

username="sa"

password="sa"

maxActive="100"

maxIdle="30"

maxWait="10000" />

2.配置你的应用程序下的 web.xml 中的之间加入:

<resource-ref>

<description>DB Connection</description>

<res-ref-name>jdbc/sample</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

3.把连接数据库的第三方驱动放到 common/lib 下面就 ok 了

4.测试程序(关键代码)

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("jdbc/sample");
Connection con = ds.getConnection();
```

以下和 JDBC 操作代码相同。。。。。。。。