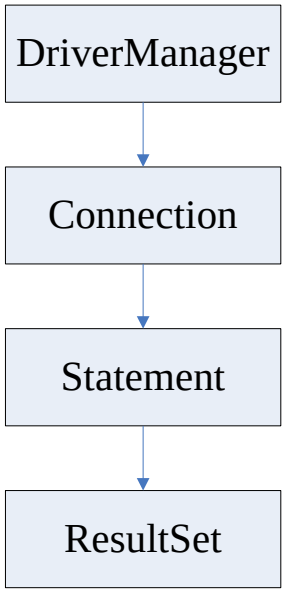
JDBC 代码手册

整理者：Eric 2008-1-13
版本号：V080113

1 JDBC 详细介绍

1.1 JDBC 类地图



- DriverManager：通过驱动，建立与数据库间的连接。
- Connection：代表着与数据库间的连接。
- Statement：代表着要执行的 SQL 语句。
- ResultSet：代表着数据库查询到的结果集。

1.2 插入数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class Run_MySql {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");
```

```
            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
           String sql = "INSERT INTO user_list ( user_name, user_password )
VALUES ( 'Eric', '123' )";

            // 执行SQL语句
            stmt = conn.createStatement();
            stmt.executeUpdate(sql);

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

1.3 修改数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class RunUpdate {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");
```

```
            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "UPDATE jdbc_teaching.user_list SET user_password
= '456' WHERE user_name = 'Eric'";

            // 执行SQL语句
            stmt = conn.createStatement();
            stmt.executeUpdate(sql);

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

1.4 删除数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class RunDelete {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 动态导入数据库的驱动
```

```
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "DELETE FROM jdbc_teaching.user_list WHERE
user_name = 'Eric'";

            // 执行SQL语句
            stmt = conn.createStatement();
            stmt.executeUpdate(sql);

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

1.5 查询数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class RunQuery {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
```

```
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
""");

            // 创造SQL语句
            String sql = "SELECT * FROM user_list";

            // 执行SQL语句
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);
            while (true) {
                // 移动到下一条数据
                boolean b = rs.next();
                // 检查下一条数据是否存在
                if ( false == b ) {
                    // 如果下一条数据不存在，就不用再遍历了
                    break;
                }
                // 取得用户名
                String userName = rs.getString( "user_name" );
                // 取得密码(字段的编号从1开始，密码排第二位)
                String userPassword = rs.getString( 2 );
                System.out.println( userName + " : " + userPassword );
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch (Exception e) {
            }
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
```

```
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

1.6 查询统计函数

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class RunQueryCount {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "SELECT COUNT(*) FROM user_list";

            // 执行SQL语句
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);

            // 移动到下一条数据
            rs.next();

            // 取得结果
            int count = rs.getInt( 1 );
```

```
            int count2 = rs.getInt( "COUNT(*)" );
            System.out.println( count + " : " + count2 );

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch (Exception e) {
            }
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

1.7 带参数的 SQL 查询

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RunQuery {

    public static void main(String[] args) {
        Connection conn = null;
        // 注意stmt的类型
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");
```

```
            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
""");

            // 创造SQL语句
            String sql = "SELECT * FROM user_list WHERE user_name = '?'";

            // 创造SQL语句
            stmt = conn.prepareStatement(sql);

            // 参数赋值(参数的顺序是从1开始的)
            stmt.setString( 1, "Eric" );

            // 执行SQL
            rs = stmt.executeQuery(sql);
            while (true) {
                // 移动到下一条数据
                boolean b = rs.next();
                // 检查下一条数据是否存在
                if ( false == b ) {
                    // 如果下一条数据不存在，就不用再遍历了
                    break;
                }
                // 取得用户名
                String userName = rs.getString( "user_name" );
                // 取得密码(字段的编号从1开始，密码排第二位)
                String userPassword = rs.getString( 2 );
                System.out.println( userName + " : " + userPassword );
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch (Exception e) {
            }
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
```

```
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

1.8 存储过程的调用

- MySql 中的存储过程：

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `test_procedure`()
BEGIN
    INSERT INTO jdbc_teaching.user_list ( user_name, user_password )
    VALUES ( 'Janet', '678' );
END
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class CallProcedure {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造调用存储过程的SQL
            String sql = "{CALL test_procedure()}";
```

```
            // 执行调用存储过程的SQL语句
            stmt = conn.prepareCall(sql);
            stmt.execute(sql);

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

2 JDBC 元数据

2.1 元数据 Meta Data

- 什么是元数据？
  - 本身固有的特性。

- 元数据分类
  - 数据库的元数据：如数据库的名称，版本等。
  - 查询结果的元数据：查询结果集中字段数量，某字段的名称等。

2.2 数据库的元数据

```
import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.Statement;

public class RunDatabaseMetaData {
```

```
    /**
     * @param args
     */
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 获取数据库元数据
            DatabaseMetaData dmd = conn.getMetaData();
            String productName = dmd.getDatabaseProductName();
            String productVersion = dmd.getDatabaseProductVersion();
            String driverName = dmd.getDriverName();
            String driverVersion = dmd.getDriverVersion();

            System.out.println(productName + " ------- " + productVersion
+ " \n"
                    + driverName + " ------- " + driverVersion);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

2.3 结果集的元数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Statement;

public class RunResultSetMetaData {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "SELECT * FROM user_list";

            // 执行SQL语句
            stmt = conn.createStatement();
            rs = stmt.executeQuery(sql);

            // 获取结果集元数据
            ResultSetMetaData rsmd = rs.getMetaData();
            int columnCount = rsmd.getColumnCount();
            String label = rsmd.getColumnLabel( 1 );

            System.out.println( columnCount + "\n" + label );

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch (Exception e) {
```

```
            }
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

3 JDBC 事务

3.1 事务的基本知识

- 由银行转帐看事务的特性
  - 原子性（Atomicity）
  - 一致性（Consistency）
  - 隔离性（Isolation）
  - 持续性（Durability）

3.2 事务实例

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class RunInsert {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
```

```
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 开启事务
            conn.setAutoCommit( false );

            // 创造SQL语句
           String sql = "INSERT INTO user_list ( user_name, user_password )
VALUES ( 'Eric', '123' )";

            // 执行SQL语句
            stmt = conn.createStatement();
            stmt.executeUpdate(sql);

            // 提交事务
            conn.commit();

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
            // 回滚事务
            try {
                conn.rollback();
            } catch ( Exception e2 ) {}
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

4 滚动 ResultSet 更新数据

4.1 ResultSet 中的 Type 取值

TYPE_FORWARD_ONLY	不可滚动
TYPE_SCROLL_INSENSITIVE	结果集可滚动，但对数据库变化不敏感
TYPE_SCROLL_SENSITIVE	结果集可滚动，但对数据库变化敏感

ResultSet 中的 Concurrency 取值

CONCUR_READ_ONLY	结果集不能用于更新数据库
CONCUR_UPDATEABLE	结果集可以用于更新数据库

4.2 修改数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class RunUpdate {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "SELECT * FROM user_list";

            // 执行SQL语句
            stmt =
```

```
conn.createStatement( ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE );
            rs = stmt.executeQuery(sql);

            while ( true ) {
                boolean b = rs.next();
                if ( false == b ) {
                    break;
                }
                String userName = rs.getString( "user_name" );
                if ( true == userName.equalsIgnoreCase( "Eric" ) ) {
                    // 更新当前数据的user_password
                    rs.updateString( "user_password", "12345678" );
                    // 提交更新
                    rs.updateRow();
                    break;
                }
            }

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch( Exception e ) {}
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

4.3 删除数据

```
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
import java.sql.Statement;

public class RunDelete {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "SELECT * FROM user_list";

            // 执行SQL语句
            stmt =
conn.createStatement( ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE );
            rs = stmt.executeQuery(sql);

            while ( true ) {
                boolean b = rs.next();
                if ( false == b ) {
                    break;
                }
                String userName = rs.getString( "user_name" );
                if ( true == userName.equalsIgnoreCase( "Eric" ) ) {
                    // 删除当前数据
                    rs.deleteRow();
                    break;
                }
            }

            System.out.println( "OK!" );
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
```

```
            // 关闭ResultSet
            try {
                rs.close();
            } catch( Exception e ) {}
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {}
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

4.4 插入数据

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class RunInsert {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        ResultSet rs = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "SELECT * FROM user_list";

            // 执行SQL语句
            stmt =
```

```
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                    ResultSet.CONCUR_UPDATABLE);
            rs = stmt.executeQuery(sql);

            while (true) {
                boolean b = rs.next();
                if (false == b) {
                    break;
                }
                String userName = rs.getString("user_name");
                if (true == userName.equalsIgnoreCase("Janet")) {
                    // 创建一条准备数据，并志向这条数据
                    rs.moveToInsertRow();
                    // 设定要插入的用户名
                    rs.updateString("user_name", "Eric");
                    // 设定要插入的密码
                    rs.updateString("user_password", "123");
                    // 提交要插入的数据
                    rs.insertRow();
                    // 指向插入前指向的那条数据
                    rs.moveToCurrentRow();
                    break;
                }
            }

            System.out.println("OK!");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch (Exception e) {
            }
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
```

```
        }
    }
}
```

5 SQL 批处理

5.1 批处理

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class RunInsert {

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 开启事务
            conn.setAutoCommit(false);

            // 创造执行SQL语句的环境
            stmt = conn.createStatement();

            // 添加处理进当前批次
            stmt
                    .addBatch("INSERT INTO user_list ( user_name,
user_password ) VALUES ('Name789', '234')");
            stmt
                    .addBatch("INSERT INTO user_list ( user_name,
user_password ) VALUES ('Name567', '234')");

            // 执行当前批次
```

```
            stmt.executeBatch();

            // 提交事务
            conn.commit();

            System.out.println("OK!");
        } catch (Exception e) {
            e.printStackTrace();
            // 回滚事务
            try {
                conn.rollback();
            } catch (Exception e2) {
            }
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

5.2 事务和批处理的区别

事务处理：底层是在数据库方存储 SQL（没有提交事务的数据放在数据库的临时表空间），最后一次把临时表空间的数据提交到数据库服务器执行实现。（消耗数据库服务器内存）

SQL 批处理：底层是在客户端把 SQL 存储起来，最后一次把客户端存储的数据发送到数据库服务器执行实现。（消耗客户端的内存）。

6 Blob 与 Clob 字段的处理

6.1 把数据写入数据库

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class RunInsertBlob {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Connection conn = null;
        // 注意这里stmt的类型
        PreparedStatement stmt = null;
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "INSERT INTO pic_list ( pic_id, pic_data ) "
                    + "VALUES ( 'pic1', ? )";

            // 将SQL语句绑定到PreparedStatement中
            stmt = conn.prepareStatement(sql);

            // 准备数据
            File file = new File("./resources/test.jpg");
            FileInputStream fis = new FileInputStream( file );

            // 设定数据
            stmt.setBinaryStream( 1, fis, ( int )file.length() );
```

```
            // 执行插入操作
            stmt.executeUpdate();

            fis.close();

            System.out.println("OK!");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭Statement
            try {
                stmt.close();
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

6.2 把数据从数据库读出来

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class RunQueryBlob {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Connection conn = null;
        // 注意这里stmt的类型
        PreparedStatement stmt = null;
        ResultSet rs = null;
```

```
        try {
            // 动态导入数据库的驱动
            Class.forName("com.mysql.jdbc.Driver");

            // 获取数据库链接
            conn = DriverManager.getConnection(
                    "jdbc:mysql://localhost:3306/jdbc_teaching", "root",
"");

            // 创造SQL语句
            String sql = "SELECT * FROM pic_list WHERE pic_id = 'pic1'";
            stmt = conn.prepareStatement(sql);

            rs = stmt.executeQuery();
            rs.next();

            // 将读到的数据写到硬盘上的管道
            File file = new File("f:\\test.jpg");
            FileOutputStream fos = new FileOutputStream(file);

            // 从数据库读数据的管道
            InputStream is = rs.getBinaryStream(2);
            int len = 0;
            byte b[] = new byte[1024];
            // 一边读一边写
            while (-1 != (len = is.read(b))) {
                fos.write(b, 0, len);
            }

            fos.close();
            is.close();

            System.out.println("OK!");
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            // 关闭ResultSet
            try {
                rs.close();
            } catch (Exception e) {
            }
            // 关闭Statement
            try {
                stmt.close();
```

```
            } catch (Exception e) {
            }
            // 关闭Connection
            try {
                conn.close();
            } catch (Exception e) {
            }
        }
    }
}
```

7 数据库连接池（JNDI）

7.1 第一步：为 Tomcat 配置连接池

%Catalina_Home%/conf/server.xml
在<host>节点中加入 Resource 配置

```
<Context path="/usermanager">          //Web 应用的根
    <Resource
            name="jdbc/tarena"                    //JNDI 名字，用于查找
            type="javax.sql.DataSource"           //资源类型
            username="root"
            password="11111111"
            driverClassName="com.mysql.jdbc.Driver" //JDBC 驱动
            maxIdle="10"
            maxWait="5000"
            url="jdbc:mysql://localhost/tarena"       //连接的 URL
            maxActive="10"/>
</Context>
```

注释：
*** maxActive**
- 同一时刻可以自数据库连接池中被分配的最大活动实例数。

*** maxIdle**
- 同一时刻数据库连接池中处于非活动状态的最大连接数。

*** maxWait**
- 当连接池中没有可用连接时，连接池在抛出异常前将等待的最大时间，单位毫秒。

7.2 第二步：在应用中配置资源引用

```
<resource-ref>
    <res-ref-name>jdbc/tarena</res-ref-name>      //资源引用名
    <res-type>javax.sql.DataSource</res-type>     //资源的类型
    <res-auth>Container</res-auth>          //Application
<res-sharing-scope>Shareable</res-sharing-scope>
      //Unshareable
</resource-ref>
```

7.3 第三步：在 Servlet 的 init 方法中通过 JNDI 接口来获取 DataSource

```
Context ctx=new InitialContext();
DataSource
ds=(DataSource)ctx.lookup("java:comp/env/jdbc/tarena");
Connection con=ds.getConnection();
```