**JDBC**

**JDBC概述**

JDBC全称Java DataBase Connectivity，可以为多种关系型数据库(DBMS) 提供统一的访问方式，用Java来操作数据库。java数据库连接，（Java Database Connectivity，简称JDBC）是Java语言中用来规范客户端程序如何来访问数据库的应用程序接口。

**JDBC常用的API（类,接口）**

DriverManager类 : 用于管理jdbc驱动

该类中常用的方法

- public static Connection getConnection(String url, String user, String

password) throws SQLException

//这是一个类方法，调用该方法可以返回一个实现connection接口的类的对象，表

明已成功连接到数据库

Connection接口: 连接数据库使用的接口

该接口中常用的方法

- Statement createStatement() throws SQLException; //返回一个Statement对象

(实现Statement接口的类的对象)

- PreparedStatement prepareStatement(String sql) throws SQLException; //返

回一个PreparedStatement对象(实现PreparedStatement接口的类的对象)，并提

前对sql语句进行预编译，通过对sql语句进行预编译，可以提高执行sql语句的效

率，还可以防止sql注入，保障了程序的安全性。

- CallableStatement prepareCall(String sql) throws SQLException; //返

回一个CallableStatement对象(实现CallableStatement接口的类的对象)

注: Statement对象，PreparedStatement对象和CallableStatement对象用于操作

数据库

Statement 接口 : 改接口的中的方法用于执行和增删改查有关的sql语句。

该接口中常用的方法

- int executeUpdate(String sql) throws SQLException; //Statement对象调用该

方法可以**执行和增删改有关的sql语句，**并返回执行sql语句受影响的行数

- ResultSet executeQuery(String sql) throws SQLException; //Statement对象调

用该方法可以**执行和查询有关的sql语句，**并返回ResultSet(查询结果集)。ResultSet

可以看成是一个Set集合，执行查询sql语句时,如果执行成功,DDMS中会显示一行

行的查询结果，每一行的查询结果都作为一个对象,添加到ResuleSet中

PreparedStatement接口: 是Statement接口的直接子接口

该接口中常用的方法

- int executeUpdate() throws SQLException; //PreparedStatement对象调

用该方法可以**执行和增删改有关的sql语句，**并返回执行sql语句受影响的行数。注

意调用该方法前执行sql语句前，必须先对sql语句进行预处理。

- ResultSet executeQuery() throws SQLException; //Statement对象调用该方法

可以**执行和查询有关的sql语句，**并返回ResultSet(查询结果集)。注意调用该方法前

执行sql语句前，必须先对sql语句进行预处理。

- void setInt(int parameterIndex, int x) throws SQLException;

- void setDouble(int parameterIndex, double x) throws SQLException;

- void setString(int parameterIndex, String x) throws SQLException;

- 等等一系列setXXX方法，不再叙述

CallableStatement接口 : 调用数据库中的存储过程/存储函数，是PreparedStatement

接口的直接子接口

ResultSet类 : 返回的结果集

**JDBC访问数据库的具体步骤**

第一步: 导入驱动(jar包)，加载具体的驱动类

第二步: 与数据库建立连接

第三步: 发送sql，执行sql语句

第四步: 处理结果集（针对查询操作)

**数据库驱动**

不同的关系型数据库对应不同的驱动，数据库驱动由开发数据库的厂商提供。常见的

关系型数据库，以及其对应的驱动如下:

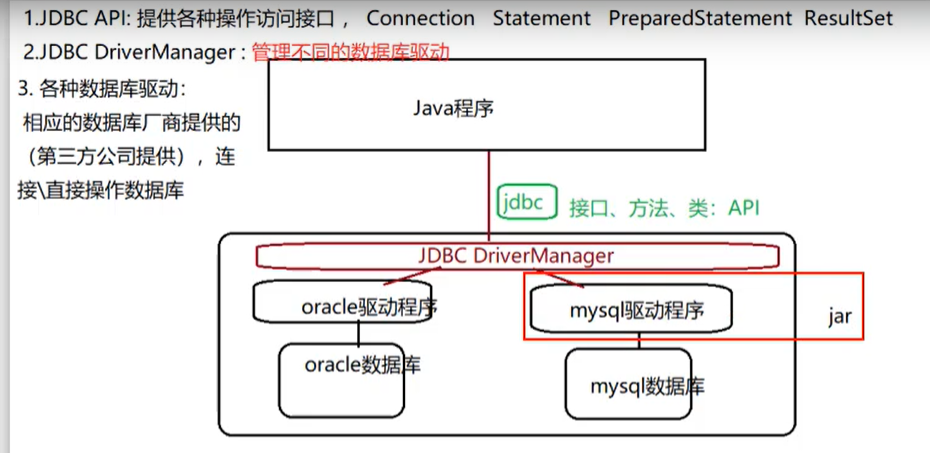
关系型数据库 驱动(一个jar包)

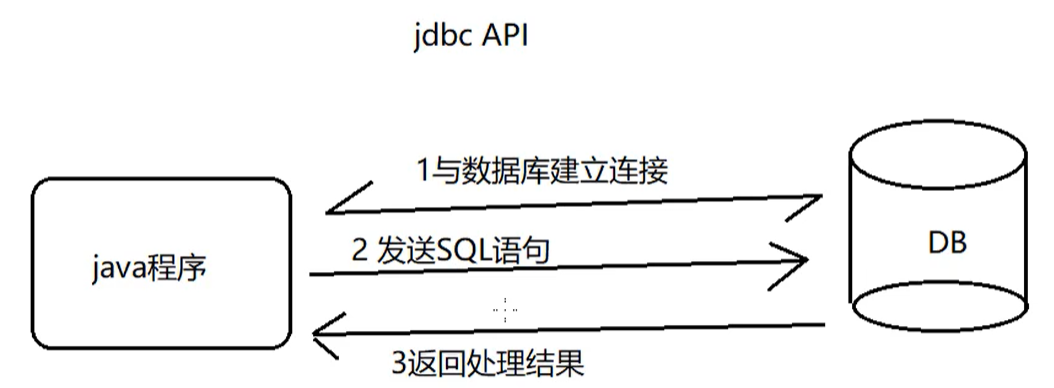
Oracle ojdbc-x.jar

MySQL mysql-connector-java-x.jar

SqlServer sqljdbc-x.jar

**java连接数据库图解**





**数据库连接池**

**背景**

直接与数据库进行连接是一项十分耗费系统资源的活动，每次连接都会花费一定的时间。一两个人同时连接数据库可能感觉不出来，但如果同时几千人连接数据库，就势必会占用很多系统资源，因此就会导致网站响应速度下降，严重话就会造成服务器崩溃。因此数据库连接池应运而生，通过使用数据库连接池，能合理的解决这个问题。

**数据库连接池概述**

数据库连接池的基本思想是: 建立一个缓冲池，预先在缓冲池中放入一定数量的连接(Connection对象)，当需要与数据库建立连接时，直接从缓冲池中取出一个连接(Connection对象)使用，用完之后在放回去。还可以通过设置连接池最大连接数来防止系统无限制的与数据进行连接，避免占用过多系统资源。

**常见的数据库连接池**

tomcat-dbcp连接池

以JNDI的方式配置数据源

第一步: 在tomcat的conf目录中的context.xml中配置数据源(dataresource)，允许配置多个数据源。代码如下:

**<resource name="student" auth="Container" type="javax.sql.DataSource"**

**maxActive="400" maxIdle="20" maxWait="5000"**

**username="root" password="haolun"**

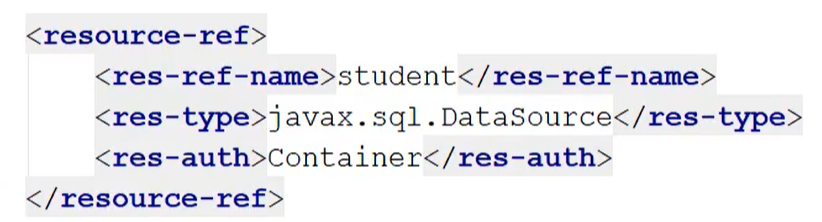
**driverClassName="com.mysql.cj.jdbc.Driver"**

**url="jdbc:mysql://localhost:3306/mydatabase" />**

注: <resource> 标签中的各属性的含义如下:



第二步: 在项目的web.xml中指定使用哪个数据源，代码如下:



第三步: 从数据源中拿数据库连接对象(connecttion对象)



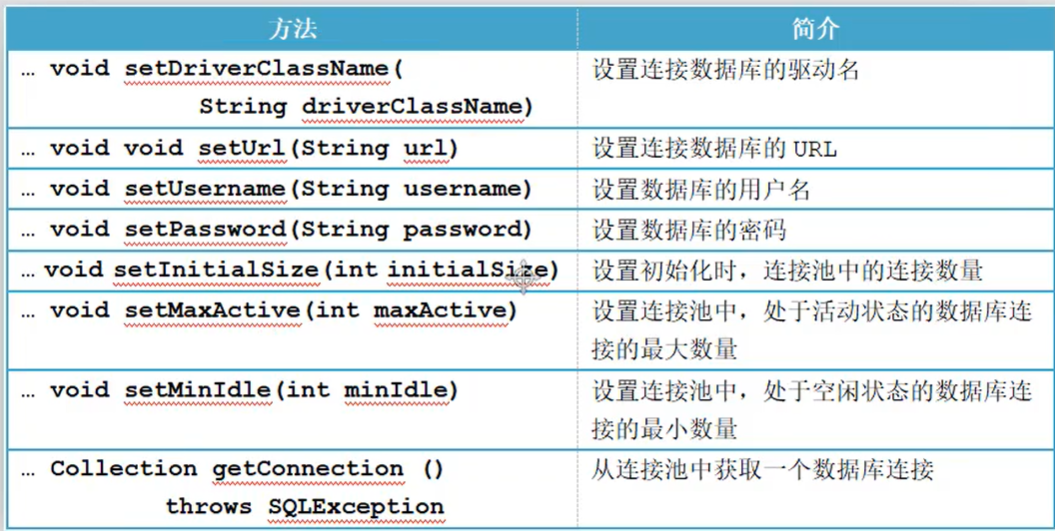
dbcp连接池

dbcp有硬编码和配置文件两种方式实现连接池，BasicDataSource类对应采用硬编码方式，BasicDataSourceFactory类对应采用配置文件。

第一步: 导入commons-dbcp-x.jar包。

注: 该jar包中主要提供BasicDataSource类、BasicDataSourceFactory类。

BasicDataSource类中常用方法



注: 发生NoClassDefFoundError异常，说明少jar包。

第二步:

方法一: 使用BasicDataSource类实现dbcp连接池（硬编码方式，即直接在类中编写代码来创建数据源，配置数据源，从连接池中获取连接）

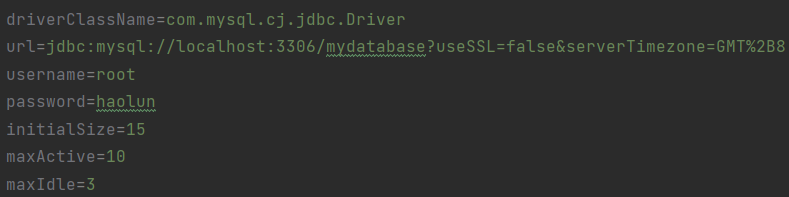
//创建数据源basicDataSource

BasicDataSource basicDataSource = new BasicDataSource();

//设置数据库驱动类名  
basicDataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");  
//设置连接数据库的URL  
basicDataSource.setUrl("jdbc:mysql://localhost:3306/mydatabase?useSSL=false&serverTimezone=GMT%2B8");  
//设置连接数据库的的用户名  
basicDataSource.setUsername("root");  
//设置连接数据库的密码  
basicDataSource.setPassword("haolun");  
//设置初始化,连接池中的"数据库连接"的数量  
basicDataSource.setInitialSize(15);  
//设置连接池中,处于活动状态的数据库连接的最大数量  
basicDataSource.setMaxActive(10);  
//设置连接池中,处于空闲状态的数据库连接的最大数量  
basicDataSource.setMaxIdle(3);  
  
try {  
 *connection* = basicDataSource.getConnection(); //从连接池中获取数据库连接  
} catch (SQLException e) {  
 e.printStackTrace();  
}

方法二: 使用BasicDataSourceFactory类实现dbcp连接池。该方法先将数据源的配置信息以key-value对的形式存放到dbcpconfig.properties属性文件中，再通过输入流读取dbcpconfig.properties属性文件，最后加载到Properties集合中。

dbcpconfig.properties（该文件要放在类路径下，也就是src目录下）

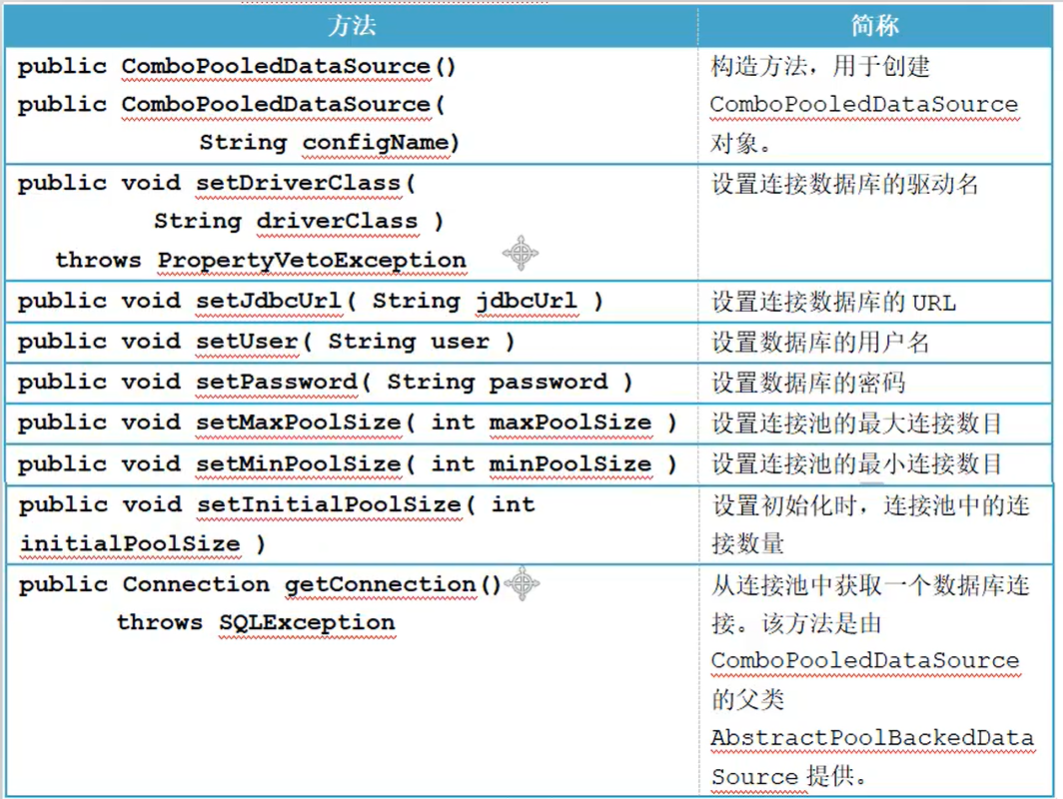




c3p0连接池

c3p0也有硬编码和配置文件两种方式实现连接池。但同dbcp不同的是，dbcp用两个不同的类来区分硬编码和配置文件，而c3p0将硬编码和配置文件方式都囊括在一个类中(ComboPooledDataSource类)。通过ComboPooledDataSource类的不同的构造方法来区分硬编码和配置文件，如果是无参数的构造方法，则使用的是硬编码方式，如果是有参数的构造方法，则使用的是配置文件方式。

ComboPooledDataSource类中常用的方法



第一步: 导入c3p0-0.9.5.2.jar和mchange-commons-java-0.2.11.jar。

第二步:

方法一: 硬链接

//使用无参数的构造方法来创建连接池,说明是硬编码方式

ComboPooledDataSource c3p0 = new ComboPooledDataSource();

//设置数据库驱动类名  
c3p0.setDriverClass("com.mysql.cj.jdbc.Driver");  
//设置连接数据库的URL  
c3p0.setJdbcUrl("jdbc:mysql://localhost:3306/mydatabase?useSSL=false&serverTimezone=GMT%2B8");  
//设置数据库用户名  
c3p0.setUser("root");  
//设置数据库密码  
c3p0.setPassword("haolun");  
//设置初始化,连接池中的"数据库连接"的数量(可不设置,有默认值)  
c3p0.setInitialPoolSize(15);  
  
//从连接池中获取数据库连接  
Connection connection = c3p0.getConnection();

方法二: 配置xml文件

配置c3p0-config.xml文件



注意: 文件名必须是c3p0-config，且必须放在类路径下，也就是src目录下，还必须是直接放到src目录下，不能放到src目录下某个包中。

//会自动从c3p0-config.xml中找name=holun的<name-config>

ComboPooledDataSource c3p0 = new ComboPooledDataSource("holun");  
//注:若想用c3p0-config.xml中的默认配置<default-config>,直接使用无参数的构造方法创建 ComboPooledDataSource对象即可  
//从连接池中获取数据库连接  
Connection connection = c3p0.getConnection();

拓展: dbcp中的数据源为BasicDataSource， c3p0中的数据源为ComboPooledDataSource。可见它们的命名格式都是xxxDataSource。需要注意的是BasicDataSource和ComboPooledDataSource都实现了javax.sql包下的DataSource接口。

druid

阿里巴巴的druid连接池在框架中用的多。