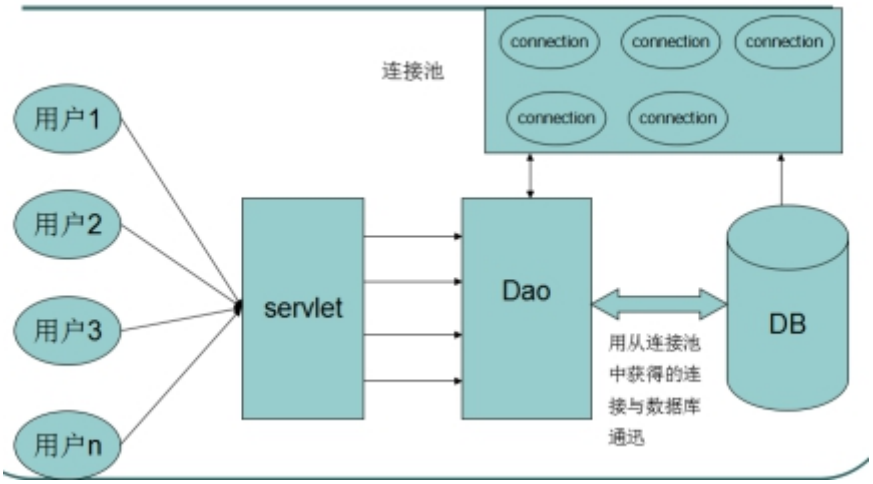
一、数据库连接池

1、连接池原理：（面试）



目的：解决建立数据库连接耗费资源和时间很多的问题，提高性能。

2、编写标准的数据源

自定义数据库连接池要实现javax.sql.DataSource接口，一般都叫数据源。

```
public class MyDataSource implements DataSource{
    //存放连接的池子
    private static LinkedList<Connection> pool = new LinkedList<Connection>();
    //创建10个连接放在池中
        static{
            for (int i = 0; i < 10; i++) {
                Connection conn = null;
                try {
                    conn = JdbcUtil.getConnection();
                    pool.addLast(conn);
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }


    public Connection getConnection() throws SQLException {
        if(pool.size()>0){
            Connection conn = pool.removeFirst();
            return conn;
        }else{
            //等待多长时间
```

3、编写数据源时遇到的问题及解决办法

```java
public class TestDataSource {
    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement ps = null;
        DataSource ds = new MyDataSource();

        try{
            conn = ds.getConnection();
            ps = conn.prepareStatement("");
//          ...
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            if(conn!=null){
                try {
                    conn.close();//关闭连接。不能关的。
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
```

a、装饰设计模式：使用频率很高

目的：改写已存在的类的某个方法或某些方法，装饰设计模式（包装模式）

口诀：

1、编写一个类，实现与被包装类相同的接口。（具备相同的行为）

2、定义一个被包装类类型的变量。

3、定义构造方法，把被包装类的对象注入，给被包装类变量赋值。

4、对于不需要改写的方法，调用原有的方法。

5、对于需要改写的方法，写自己的代码。

```java
//4、对于不需要改写的方法，调用原有的方法。
//5、对于需要改写的方法，写自己的代码。
public class MyConnection implements Connection {

    private Connection oldConn;
    LinkedList<Connection> pool;
    public MyConnection(Connection oldConn,LinkedList<Connection> pool){
        this.oldConn = oldConn;
        this.pool = pool;
    }

    public void close() throws SQLException {
        pool.add(oldConn);
    }

    public Statement createStatement() throws SQLException {
        return oldConn.createStatement();
    }

    @Override
    public <T> T unwrap(Class<T> iface) throws SQLException {
        return oldConn.unwrap(iface);
    }
```

b、默认适配器：装饰设计模式一个变体

```
//本身就是一个装饰类，对原类没有任何改变。
//1、编写一个类，实现与被包装类相同的接口。（具备相同的行为）
//2、定义一个被包装类类型的变量。
//3、定义构造方法，把被包装类的对象注入，给被包装类变量赋值。
//4、对于不需要改写的方法，调用原有的方法。
public class ConnectionWarper implements Connection {

    private Connection oldConn;
    public ConnectionWarper(Connection oldConn){
        this.oldConn = oldConn;
    }
    @Override
    public <T> T unwrap(Class<T> iface) throws SQLException {
        return oldConn.unwrap(iface);
    }

    @Override
    public boolean isWrapperFor(Class<?> iface) throws SQLException {
        return oldConn.isWrapperFor(iface);
    }
```

```
//1、编写一个类，继承包装类适配器。（具备相同的行为）
//2、定义一个被包装类类型的变量。
//3、定义构造方法，把被包装类的对象注入，给被包装类变量赋值。
//4、对于不需要改写的方法，调用原有的方法。
public class MyConnection1 extends ConnectionWarper{

/*  public MyConnection1(){
        super();
    }*/
    private Connection conn;
    public MyConnection1(Connection conn){
        super(conn);
        this.conn = conn;
    }
    public void close() throws SQLException {
//      ... 写自己的代码
    }
```

3、常用的数据源配置（日后都使用数据源，一定要配置一下）

3.1、DBCP

DBCP：Apache推出的Database Connection Pool

使用步骤：

> 添加jar包　commons-dbcp-1.4.jar commons-pool-1.5.6.jar

> 添加属性资源文件

> 编写数据源工具类

```
public class DBCPUtils {
    //声明一个连接池对象
    private static DataSource ds;
    static{
        Properties p = new Properties();
        try {
            p.load(DBCPUtils.class.getClassLoader().getResourceAsStream("dbcpconfig.properties"));
            ds = BasicDataSourceFactory.createDataSource(p);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    //得到Connection对象
    public static Connection getConnection(){
        try {
            return ds.getConnection();
        } catch (SQLException e) {
            throw new RuntimeException("创建数据库连接失败");
        }
    }
     * 释放资源
    public static void release(ResultSet rs,Statement stmt,Connection conn){
```

3.2、C3P0

使用步骤：

1、添加jar包

2、编写配置文件

c3p0-config.xml，放在classpath中，或classes目录中

```
<?xml version="1.0" encoding="UTF-8"?>
<c3p0-config>
    <default-config>
        <property name="driverClass">com.mysql.jdbc.Driver</property>
        <property name="jdbcUrl">jdbc:mysql://localhost:3306/day05</property>
        <property name="user">root</property>
        <property name="password">root</property>

        <property name="initialPoolSize">10</property>
        <property name="maxIdleTime">30</property>
        <property name="maxPoolSize">100</property>
        <property name="minPoolSize">10</property>
        <property name="maxStatements">200</property>
    </default-config>

</c3p0-config>
```

3、编写工具类：

```
public class C3P0Util {
    private static ComboPooledDataSource datasource = new ComboPooledDataSource();

    //得到Connection对象
    public static Connection getConnection(){
        try {
            return datasource.getConnection();
        } catch (SQLException e) {
            throw new RuntimeException("创建数据库连接失败");
        }
    }
     * 释放资源
    public static void release(ResultSet rs,Statement stmt,Connection conn){
```

5、用JavaWeb服务器管理数据源：Tomcat

开发JavaWeb应用，必须使用一个JavaWeb服务器，JavaWeb服务器都内置数据源。

Tomcat：（DBCP）

数据源只需要配置服务器即可。

配置数据源的步骤：

1、拷贝数据库连接的jar到tomcatlib目录下

2、配置数据源XML文件

a)如果把配置信息写在tomcat下的conf目录的context.xml中，那么所有应用都能使用此数据源。

b)如果是在当前应用的META-INF中创建context.xml,编写数据源，那么只有当前应用可以使用。

```
<Resource name="jdbc/day13" auth="Container" type="javax.sql.DataSource"
              maxActive="100" maxIdle="30" maxWait="10000"
              username="root" password="root" driverClassName="com.mysql.jdbc.Driver"
              url="jdbc:mysql://localhost:3306/day05"/>
```

3、使用连接池

```
<%
    Context ctx = new InitialContext();
    DataSource ds = (DataSource)ctx.lookup("java:/comp/env/jdbc/day13");
    out.print(ds);
%>
```

path + name

JNDI：java nameing directory interface

JNDI容器就是一个Map

key（String）	value（Object）
path+name	对象
path+"jdbc/day16"	DataSource对象

二、自定义JDBC框架（练技术）

1、数据库元信息的获取（很简单、很无聊、很重要）

a、元信息：（Meta Data）指数据库或表等的定义信息。

```java
//DatabaseMetaData: DBMS的元信息
@Test
public void test1() throws Exception{
    Connection conn = C3P0Util.getConnection();
    DatabaseMetaData dmd = conn.getMetaData();
    System.out.println(dmd.getDatabaseProductName()+":"+dmd.getDatabaseProductVersion());
    conn.close();
}
//ParameterMetaData:预制语句中的SQL占位符元信息
@Test
public void test2() throws Exception{
    Connection conn = C3P0Util.getConnection();
    PreparedStatement stmt = conn.prepareStatement("??????????");
    ParameterMetaData pmd = stmt.getParameterMetaData();
    int count = pmd.getParameterCount();//只统计?的个数
    System.out.println("语句中有几个占位符: "+count);
    stmt.close();
    conn.close();
}
//ResultSetMetaData:结果集元数据信息
@Test
public void test3() throws Exception{
    Connection conn = C3P0Util.getConnection();
    PreparedStatement stmt = conn.prepareStatement("select * from account");
    ResultSet rs = stmt.executeQuery();
    ResultSetMetaData rsmd = rs.getMetaData();
    int count = rsmd.getColumnCount();
```

2、自定义JDBC框架

反射；策略设计模式；