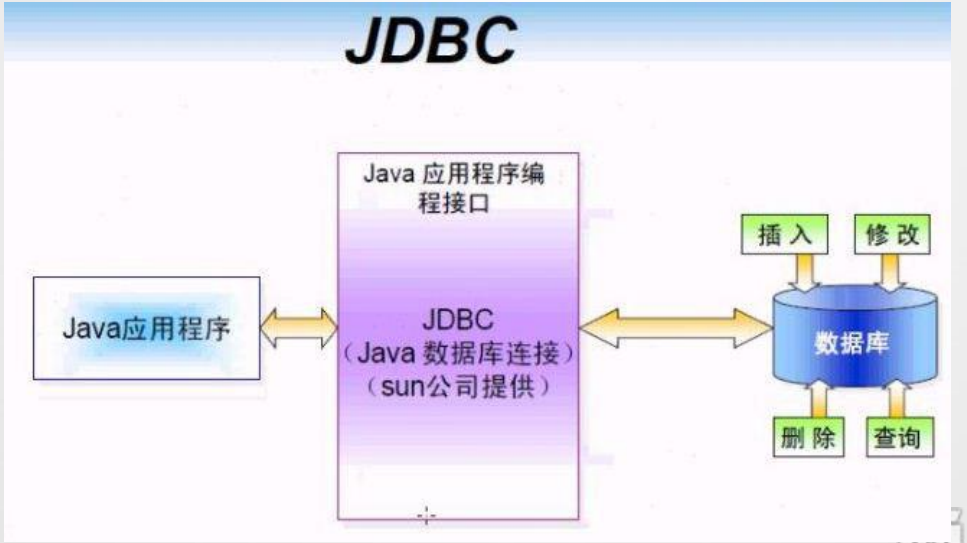
8. JDBC

CONTENTS

- JDBC概述
- 使用JDBC完成添加/更新/删除操作
- 使用JDBC完成查询操作
- JDBC语法总结
- 使用PreparedSatement完善JDBC操作
- 手动启动事务管理
- 采用分层实现JDBC案例

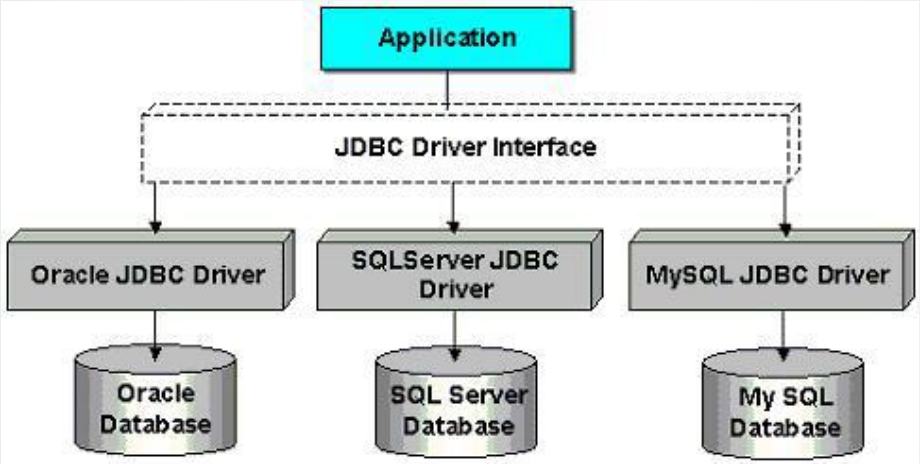
JDBC概述

- 什么是JDBC
  - JDBC（Java Data Base Connectivity,Java数据库连接）
  - 是一种用于执行SQL语句的Java API，为多种关系数据库提供统一访问
  - 它由一组用Java语言编写的类和接口组成



JDBC概述

- 有了JDBC，程序员只需用JDBC API写一个程序，就可以访问所有数据库。



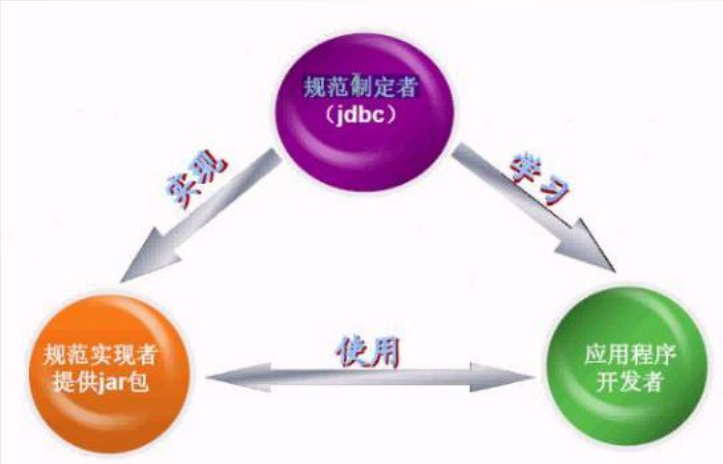
- 将Java语言和JDBC结合起来使程序员不必为不同的平台编写不同的应用程序，只须写一遍程序就可以让它在任何平台上运行，这也是Java语言“编写一次，处处运行”的优势。

JDBC概述

- JDBC API
  - 提供者：Sun公司
  - 内容：供程序员调用的接口与类，集成在java.sql和javax.sql包中，如
    - DriverManager类 作用：管理各种不同的JDBC驱动
    - Connection接口
    - Statement接口
    - ResultSet接口
- JDBC 驱动
  - 提供者：数据库厂商
  - 作用：负责连接各种不同的数据库
- JDBC对Java程序员而言是API，对实现与数据库连接的服务提供商而言是接口模型。

JDBC概述

- 三方关系
  - SUN公司是规范制定者，制定了规范JDBC（连接数据库规范）
  - 数据库厂商微软、甲骨文等分别提供实现JDBC接口的驱动jar包
  - 程序员学习JDBC规范来应用这些jar包里的类。



JDBC概述

- JDBC访问数据库步骤
  - 1：加载一个Driver驱动
  - 2：创建数据库连接（Connection）
  - 3：创建SQL命令发送器Statement
  - 4：通过Statement发送SQL命令并得到结果
  - 5：处理结果（select语句）
  - 6：关闭数据库资源
    - ResultSet
    - Statement
    - Connection。

JDBC语法总结

- 1.加载驱动
  - 加载JDBC驱动是通过调用方法java.lang.Class.forName()，下面列出常用的几种数据库驱动程序加载语句的形式：
    - Class.forName(“oracle.JDBC.driver.OracleDriver”)；//使用Oracle的JDBC驱动程序
    - Class.forName(“com.microsoft.JDBC.sqlserver.SQLServerDriver”)；//使用SQL Server的JDBC驱动程序
    - Class.forName(“com.ibm.db2.JDBC.app.DB2Driver”)；//使用DB2的JDBC驱动程序
    - Class.forName("com.mysql.JDBC.Driver");//使用MySql的JDBC驱动程序

JDBC语法总结

- 2.创建数据库连接
  - 与数据库建立连接的方法是调用DriverManager.getConnection(String url, String user, String password)方法

```
Connection conn=null;
String url="jdbc:oracle:thin:@localhost:1521:orcl";
String user=“scott";
String password=“tiger";
conn = DriverManager.getConnection(url, user, password);
```

JDBC语法总结

- 3.创建Statement并发送命令
  - Statement对象用于将 SQL 语句发送到数据库中，或者理解为执行sql语句
  - 有三种 Statement对象：
    - Statement：用于执行不带参数的简单SQL语句；
    - PreparedStatement（从 Statement 继承）：用于执行带或不带参数的预编译SQL语句；
    - CallableStatement（从PreparedStatement 继承）：用于执行数据库存储过程的调用。

方法	作用
ResultSet executeQuery(String sql)	可以执行插入、删除、更新等操作，返回值是执行该操作所影响的行数
boolean execute(String sql)	可以执行任意SQL语句，然后获得一个布尔值，表示是否返回ResultSet
int executeUpdate(String sql)	执行SQL查询并获取到ResultSet对象

JDBC语法总结

- 4.处理ResultSet结果
  - ResultSet对象是executeQuery()方法的返回值，它被称为结果集，它代表符合SQL语句条件的所有行，并且它通过一套getXXX方法（这些get方法可以访问当前行中的不同列）提供了对这些行中数据的访问。
  - ResultSet里的数据一行一行排列，每行有多个字段，且有一个记录指针，指针所指的数据行叫做当前数据行，我们只能来操作当前的数据行。我们如果想要取得某一条记录，就要使用ResultSet的next()方法 ,如果我们想要得到ResultSet里的所有记录，就应该使用while循环。
  - ResultSet对象自动维护指向当前数据行的游标。每调用一次next()方法，游标向下移动一行。
  - 初始状态下记录指针指向第一条记录的前面，通过next()方法指向第一条记录。循环完毕后指向最后一条记录的后面。

JDBC语法总结

- 4.处理ResultSet结果

方法名	说　　明
boolean next()	将光标从当前位置向下移动一行
boolean previous()	游标从当前位置向上移动一行
void close()	关闭ResultSet 对象
int getInt(int colIndex)	以int形式获取结果集当前行指定列号值
int getInt(String colLabel)	以int形式获取结果集当前行指定列名值
float getFloat(int colIndex)	以float形式获取结果集当前行指定列号值
float getFloat(String colLabel)	以float形式获取结果集当前行指定列名值
String getString(int colIndex)	以String 形式获取结果集当前行指定列号值
String getString(String colLabel)	以String形式获取结果集当前行指定列名值

JDBC语法总结

- 5.关闭数据库资源
  - 作为一种好的编程风格，应在不需要Statement对象和Connection对象时显式地关闭它们。关闭Statement对象和Connection对象的语法形式为：
  - public void close() throws SQLException

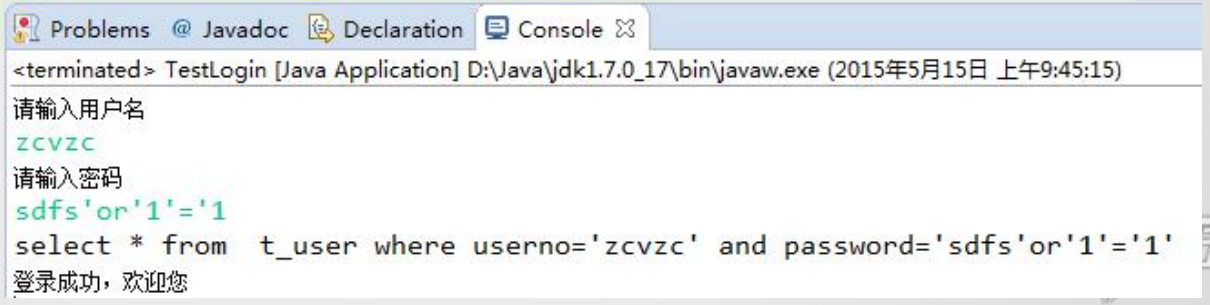
  - 用户不必关闭ResultSet。当它的 Statement 关闭、重新执行或用于从多结果序列中获取下一个结果时，该ResultSet将被自动关闭。

  - 注意：要按先ResultSet结果集，后Statement，最后Connection的顺序关闭资源，因为Statement和ResultSet是需要连接是才可以使用的，所以在使用结束之后有可能其他的Statement还需要连接，所以不能先关闭Connection。

使用PreparedSatement完善JDBC操作

- 业务：输入用户名和密码，判断登录是否成功
- 缺陷：存在SQL注入风险

```
stmt = conn.createStatement();
String sql="select * from  t_user where userno='"
                    +userno+"' and password='"+upwd+"'";
rs = stmt.executeQuery(sql);
System.out.println(sql);
if(rs.next()){
System.out.println("登录成功，欢迎您");
}else{
System.out.println("用户名或者密码错误，请重新登录");
}
```



使用PreparedSatement完善JDBC操作

- PreparedStatement 接口继承 Statement接口
- 如果需要多次执行一个SQL语句，可以使用PreparedStatement对象。在创建PreparedStatement对象时，通过传递不同参数值多次执行PreparedStatement对象，可以得到多个不同的结果。

- 优势：执行效率高、代码可读性强、安全性高

- 该对象用Connection的prepareStatement()方法创建。如：
  - pstmt=conn.prepareStatement("insert into student values(?,?,?)");
  - pstmt.setString(1, “小明");
  - pstmt.setInt(2, 27);
  - pstmt.setFloat(3, 85);
  - pstmt.executeUpdate();

使用PreparedSatement完善JDBC操作

- PreparedStatement接口中的主要方法
  - void setString (int parameterIndex, String x)
  - void setFloat (int parameterIndex, float x)
  - void setInt (int parameterIndex, int x)
  - void setDate (int parameterIndex, java.sql.Date x)
  - void setDouble (int parameterIndex, double x)

  - ResultSet executeQuery () //返回单结果集，通常用于SELECT语句
  - boolean execute () //返回布尔值，通常用于insert，update，delete语句
  - int executeUpdate () //返回操作影响的行数，通常用于insert，update，delete语句

手动启动事务管理

- 在JDBC中，事务操作缺省是自动提交。
  - 一条对数据库的更新表达式代表一项事务操作
  - 操作成功后，系统将自动调用commit()提交，否则调用rollback()回滚
- 在JDBC中，事务操作方法都位于接口java.sql.Connection中
  - 可以通过调用setAutoCommit(false)来禁止自动提交。
  - 之后就可以把多个数据库操作的表达式作为一个事务，在操作完成后调用commit()来进行整体提交，
  - 倘若其中一个表达式操作失败，都不会执行到commit()，并且将产生响应的异常；
  - 此时就可以在异常捕获时调用rollback()进行回滚,回复至数据初始状态
- 事务结束的边界是commit或者rollback方法的调用
- 事务开始的边界则不是那么明显了，它会开始于组成当前事务的所有statement中的第一个被执行的时候。

手动启动事务管理

```
try {
        stmt = conn.createStatement();
        conn.setAutoCommit(false);
        String sql1 ="update account set balance = balance-1000 where aid=1";
        String sql2 ="update account set balance = balance+1000 where aid=2";
         stmt.executeUpdate(sql1);
         stmt.executeUpdate(sql2);
         conn.commit();
} catch (SQLException e) {
        try {
            conn.rollback();
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    e.printStackTrace();
}
```

采用分层实现JDBC案例

- 完成对雇员数据的多种操作
  - 查询所有雇员
  - 按照编号查询雇员
  - 添加雇员
  - 删除雇员
- 具体实现
  - 定义包结构
  - 定义实体类Employee
  - 定义EmployeeDao接口
  - 定义EmployeeDaoImpl
  - 抽取BaseDao
  - 测试

作业

- 使用JDBC的基本过程是什么？

- 建立一个student表，包含学号id、姓名name、年龄age、生日birthday，系别department。编写java类接收来自客户端的学生信息，并存入数据库表中；把数据库表中所有记录输出到屏幕。

- 建立一个books表，包含书号bookid、书名name、出版社press、作者author，价格price和出版日期pdate字段。编写Java类实现对books表的分别按书号、书名、作者的查询，并且实现信息的插入和删除。