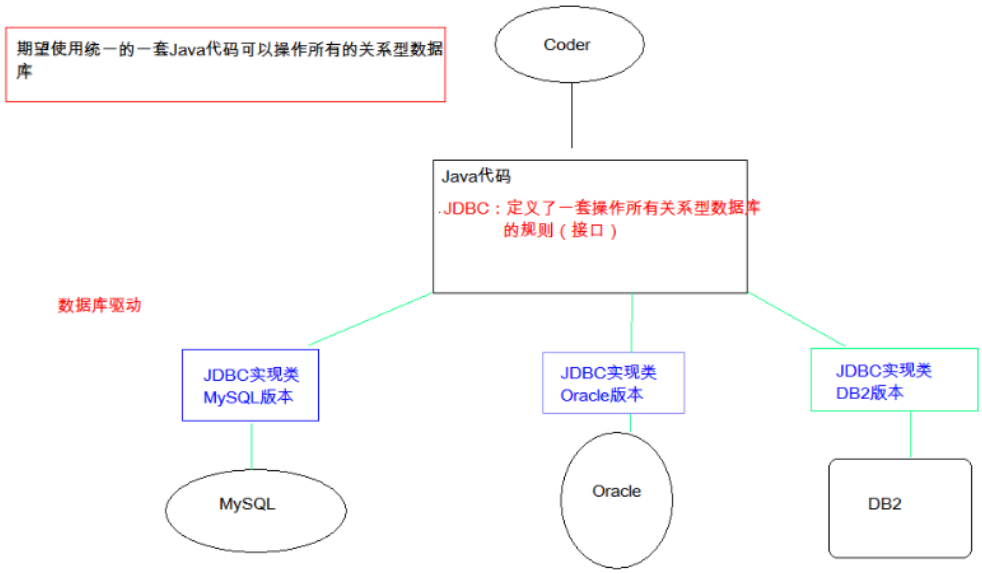
JDBC

1. 基本概念

- Java DataBase Connection，Java数据库连接；定义了操作所有关系型数据库的规范（接口）。本质是官方公司定义的一套操作所有关系型数据库的规则，各个数据库厂商去实现这套接口，提供数据库驱动jar包。



2. 执行步骤

- 导入驱动jar包：
- 注册驱动
- 获取数据库连接对象connection
- 定义sql
- 获取执行sql语句的对象statement
- 执行sql，接受返回结果
- 处理结果
- 释放资源

```
//代码实现：
        //1. 导入驱动jar包
        //2.注册驱动
        Class.forName("com.mysql.jdbc.Driver");
        //3.获取数据库连接对象
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/db3", "root", "root");
        //4.定义sql语句
        String sql = "update account set balance = 500 where id = 1";
```

```
        //5.获取执行sql的对象 Statement
        Statement stmt = conn.createStatement();
        //6.执行sql
        int count = stmt.executeUpdate(sql);
        //7.处理结果
        System.out.println(count);
        //8.释放资源
        stmt.close();
        conn.close();
```

3. JDBC各个对象：

- DriverManager：
  1. 注册驱动：告诉程序该使用哪一个数据库驱动jar，mysql5之后的驱动jar包可以省略注册驱动的步骤。
  2. 获取数据库连接:
     - 方法：static Connection getConnection(String url, String user, String password)
     - 参数：
       i. url：指定连接的路径。语法：jdbc:mysql://ip地址(域名):端口号/数据库名称
       ii. user：用户名
       iii. password：密码
- Connection：数据库连接对象
  1. 获取执行sql的对象
     分为两种：
     - Statement createStatement() 可执行静态的SQL语句
     - PreparedStatement prepareStatement(String sql) 可执行预编译的SQL语句
  2. 管理事务
     - 开启事务：
       setAutoCommit(boolean autoCommit)：
       调用该方法设置参数为false，即开启事务
     - 提交事务：commit()
     - 回滚事务：rollback()
- Statement：执行sql的语句
  1. 用于执行静态sql语句并返回其生成结果的对象

boolean execute(String sql)	执行任意的sql
int executeUpdate(String sql)	执行DML语句、DDL语句 返回值为影响的行数
ResultSet executeQuery(String sql)	执行DQL（select)语句

- ResultSet：结果集对象，封装查询的结果

boolean next()	游标向下移动一行，判断当前行是否是最后一行末尾(是否有数据)，如果是，则返回false，如果不是则返回true
getXxx( ) Xxx表示类型，可以为数据类型	参数： 1. int：代表列的编号,从1开始 如： getString(1) 2. String：代表列名称。 如： getDouble(”balance”)

- PreparedStatement：

1. sql注入：在拼接sql语句时，有一些sql的特殊关键字参与字符串的拼接，造成安全性的问题；为了解决sql注入的问题，采用preparedStatement解决。
2. 预编译的sql：参数使用？作为占位符
3. 优点：
   a. 防止sql注入问题
   b. 效率更高

4. JDBC管理事务

1. 事务：一个包含多个步骤的业务操作。如果这个业务操作被事务管理，则这多个步骤要么同时成功，要么同时失败。
2. 操作：开启事务 -> 提交事务 -> 如果失败后回滚事务

5. 数据库连接池

1. 概念：其实就是一个容器(集合)，存放数据库连接的容器。当系统初始化好后，容器被创建，容器中会申请一些连接对象，当用户来访问数据库时，从容器中获取连接对象，用户访问完之后，会将连接对象归还给容器。
2. 优点：节约创建连接的资源；用户访问更加高效
3. 实现：基于Datasource实现；一般由数据库厂商实现；常见的有CP30和Druid。
   - CP30：

     使用步骤：

     1. 导入jar包 (两个) c3p0-0.9.5.2.jar->mchange-commons-java-0.2.12.jar ，
     2. 定义配置文件：名称为c3p0.properties 或者 c3p0-config.xml，存放路径为src目录下。
     3. 创建核心对象 数据库连接池对象 ComboPooledDataSource
     4. 获取连接： getConnection

   - Druid：数据库连接池实现技术，由阿里巴巴提供的
     1. 步骤：
        1. 导入jar包 druid-1.0.9.jar
        2. 定义配置文件：
           i. 是properties形式的
           ii. 可以叫任意名称，可以放在任意目录下
        3. 加载配置文件。Properties
        4. 获取数据库连接池对象：通过工厂来来获取 DruidDataSourceFactory
        5. 获取连接：getConnection

6. JDBCTemplate

1. Spring框架对JDBC的简单封装，提供了一个JDBCTemplate对象简化JDBC的开发。
2. 使用步骤：
   a. 导入jar包
   b. 创建JdbcTemplate对象。依赖于数据源DataSource

```
JdbcTemplate template = new JdbcTemplate(ds);
```

   c. 调用JdbcTemplate的方法来完成CRUD的操作

update()	执行DML语句（增、删、改语句）
queryForMap()	将结果集封装为map集合，将列名作为key，将值作为value。将这条记录封装为一个map集合。 注意：这个方法查询的结果集长度只能是1
queryForList()	查询结果将结果集封装为list集合

query()	查询结果，将结果封装为JavaBean对象 一般我们使用BeanPropertyRowMapper实现类，可以完成数据到JavaBean的自动封装 new BeanPropertyRowMapper<类型>(类型.class)
queryForObject	查询结果，将结果封装为对象。一般用于聚合函数的查询