Java的JDBC编程

> ## 本节目标
>
> 1.掌握JDBC的概念与使用场景 2.掌握JDBC的工作原理 3.掌握JDBC中几个常用接口和类 4.掌握基于数据库的应用程序开发流程

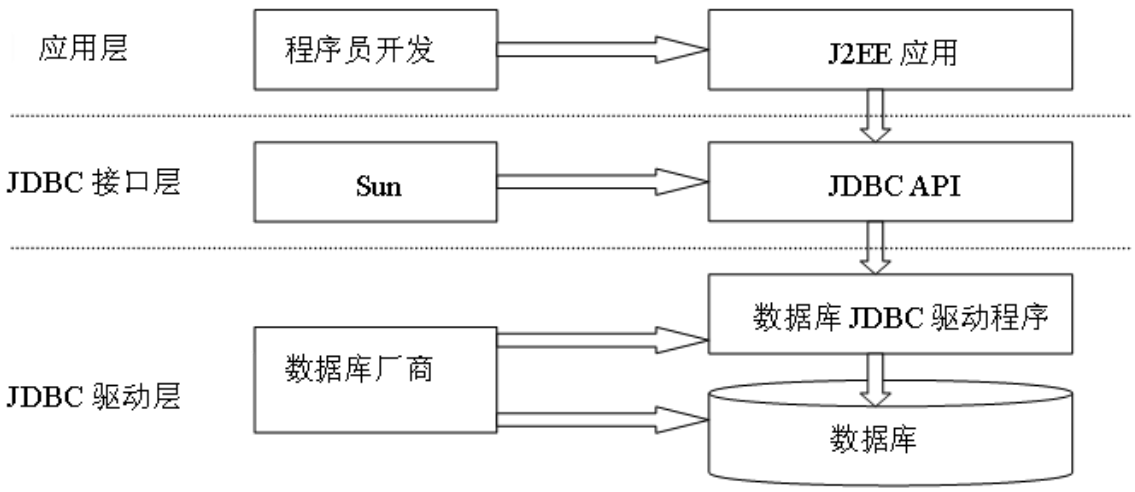
1. 什么是JDBC

现在的应用系统大多都离不开数据库，Java程序访问数据库的基本方式是通过JDBC。JDBC（Java DataBase Connectivity，Java数据库连接）技术的简称，是一种用于执行SQL语句的Java API。它由一组用Java编程语言编写的类和接口组成。这个API由`java.sql.*,javax.sql.*`包中的一些类和接口组成，它为数据库开发人员提供了一个标准的API，使它们能够用纯Java API 来编写数据库应用程序。

2. JDBC工作原理

JDBC 为多种关系数据库提供了统一访问方式，作为特定厂商数据库访问API的一种高级抽象，它主要包含一些通用的接口类。真正的数据库访问操作实现是由各自数据库厂商提供的。通常把厂商提供的特定于数据库的访问API称为数据库JDBC驱动程序。

JDBC访问数据库层次结构:

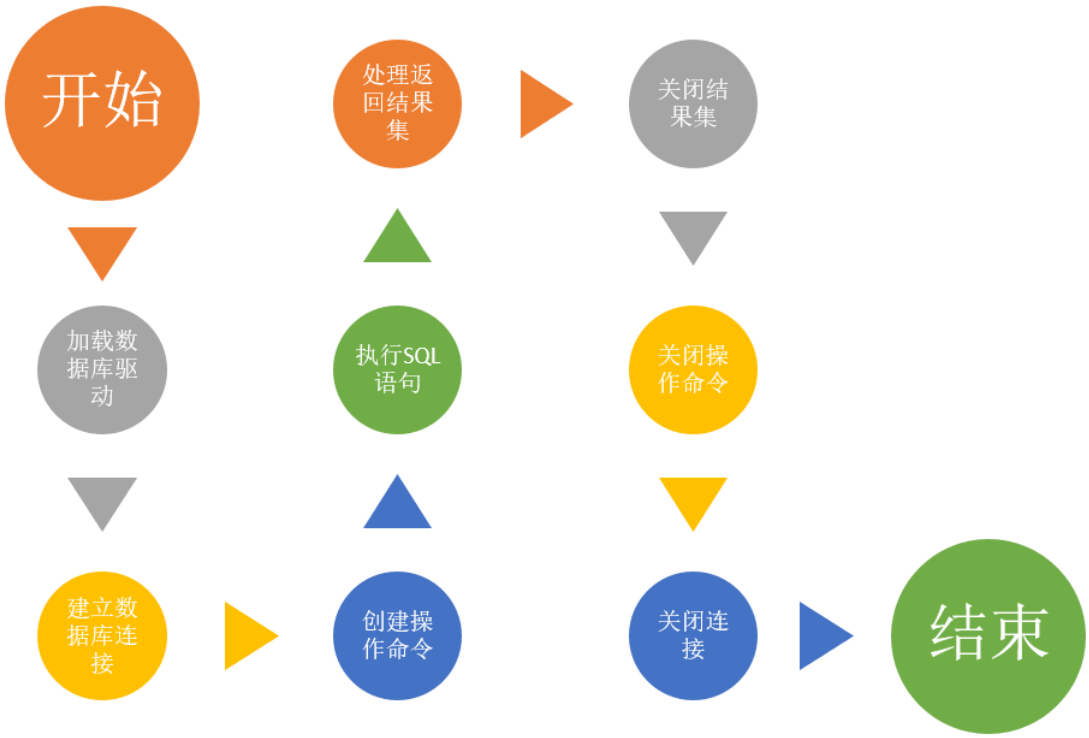


JDBC优势：

- Java语言访问数据库操作完全面向抽象接口编程
- 开发数据库应用不用限定在特定数据库厂商的API
- 程序的可移植性大大增强

3. JDBC使用步骤

3.1 JDBC使用流程图



Java JDBC 编程操作步骤

3.2 JDBC开发案例

- 准备数据库环境（创建一个便签应用的数据库以及表）

```
drop database if exists `memo`;
create database if not exists `memo` default character set utf8 collate
utf8_general_ci;

use `memo`;

drop table if exists `memo_group`;
create table if not exists `memo_group`(
    id int primary key auto_increment comment '便签组编号',
    name varchar(8) not null unique key comment '便签组名称',
    created_time datetime not null comment '创建时间',
    modify_time timestamp comment '修改时间'
)engine innodb;

-- 默认分组
insert into `memo_group` (id,name,created_time) values (1, '默认组', now());

drop table if exists `memo_info`;
create table if not exists `memo_info`(
    id int primary key auto_increment comment '便签编号',
    group_id int not null comment '便签组编号',
    title varchar(32) not null comment '便签标题',
    content varchar(1024) not null default '' comment '便签内容',
```

```
    is_protected char(1) not null default '0' comment '是否私密，0：公开 1：私密',
    background enum('WHITE','RED','BLUE','GREEN') default 'WHITE' comment '背景颜
色',
    is_remind  char(1) default '0' comment '是否提醒，0：不提醒 1：提醒',
    remind_time datetime comment '提醒时间',
    created_time datetime not null comment '创建时间',
    modify_time timestamp comment '修改时间'
)engine innodb;

-- 欢迎标签
insert into `memo_info`(id,group_id,title,content,created_time) values (1,1,'欢迎使
用','下面是使用手册',now());

drop table if exists `memo_share`;
create table if not exists `memo_share`(
    id int primary key auto_increment comment '便签分享编号',
    info_id int comment '便签编号',
    mark varchar(32) not null default '' comment '分享备注',
    share_time  datetime not null comment '分享时间'
)engine innodb;

-- 欢迎标签分享
insert into `memo_share`(id,info_id,mark, share_time) values (1,1,'特别有意思的便签
APP',now());
```

- 准备数据库驱动包（比如: MySQL数据库的驱动包）
- 加载JDBC驱动程序

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

- 建立数据库连接

```
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/memo?
user=root&password=root&useUnicode=true&characterEncoding=UTF-8");
```

```
//MySQL数据连接的URL参数格式如下：
jdbc:mysql://服务器地址:端口/数据库名?参数名=参数值
```

- 创建操作命令（statement）

```
Statement statement = connection.createStatement();
```

- 执行SQL语句

```
ResultSet resultSet= statement.executeQuery(
                "select id,group_id,title,content,is_protected,
background,is_remind,remind_time,created_time,modify_time from memo_info");
```

- 处理结果集

```
while (resultSet.next()) {
    int id = resultSet.getInt("id");
    String title = resultSet.getString("title");
    String content = resultSet.getString("content");
    Date createTime = resultSet.getDate("created_time");
    System.out.println(String.format("Memo: id=%d, title=%s, content=%s,
createTime=%s", id, title, content, createTime.toString()));
}
```

- 释放资源（关闭结果集，命令，连接）

```
//关闭结果集
if (resultSet != null) {
    try {
        resultSet.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
//关闭命令
if (statement != null) {
    try {
        statement.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
//关闭连接命令
if (connection != null) {
    try {
        connection.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

4. JDBC常用接口和类

4.1 JDBC API

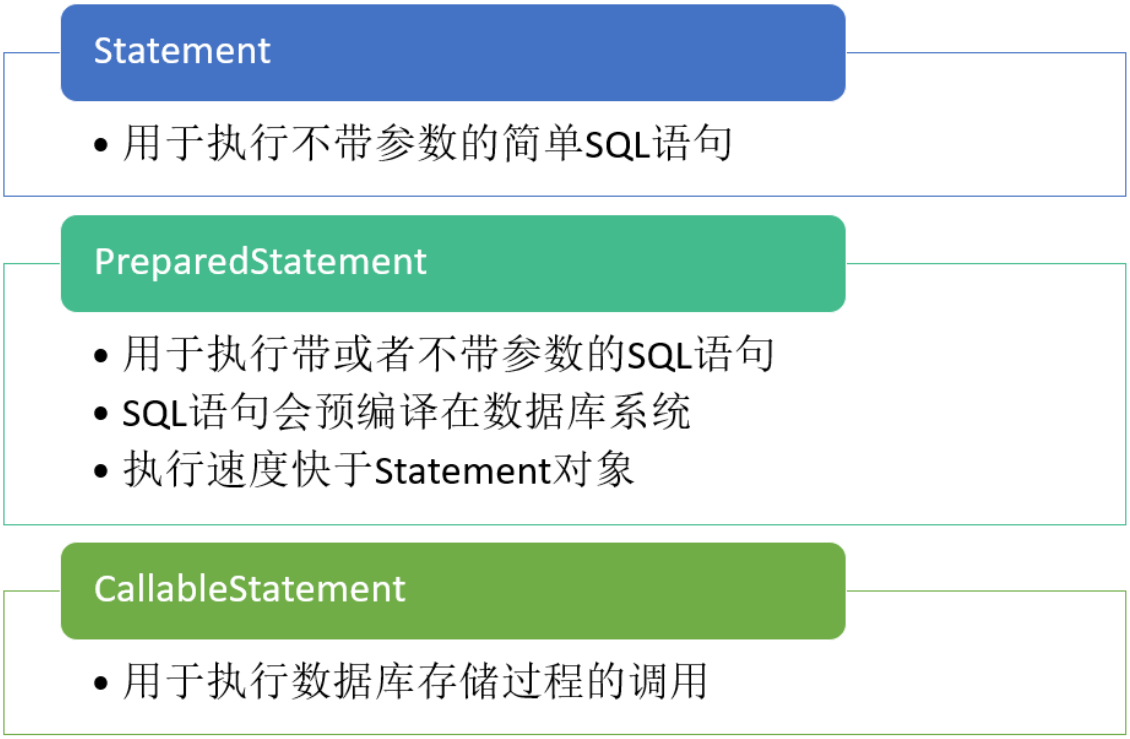
在Java JDBC编程中对数据库的操作均使用JDK自带的API统一处理，通常与特定数据库的驱动类是完全解耦的。所以掌握Java JDBC API （位于`java.sql`包下） 即可掌握Java数据库编程。

4.1 数据库连接

Connection接口实现类由数据库提供，获取Connection对象通常有两种方式：一种是通过DriverManager的静态方法获取，一种是通过DataSource（数据源）对象获取。**实际应用中会使用DataSource对象。**

4.2 Statement对象

Statement对象主要是将SQL语句发送到数据库中。JDBC API中主要提供了三种Statement对象。



实际开发中最常用的是PreparedStatement对象，以下对其的总结：

主要掌握两种执行SQL的方法：

- executeQuery() 方法执行后返回单个结果集的，通常用于select语句
- executeUpdate()方法返回值是一个整数，指示受影响的行数，通常用于update、insert、delete语句

4.3 ResultSet对象

ResultSet对象它被称为结果集，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，并且有一个记录指针，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。

5. 便签应用案例

技术知识点：

- JDBC API的CRUD

- JDBC API的事务控制

功能要求：

- 便签组
  - 创建便签组
  - 修改便签组
  - 删除便签组
  - 查询指定组中的便签
- 便签
  - 创建便签
  - 根据编号删除便签
  - 根据时间查询便签
  - 根据内容查询便签
  - 分页查询便签
  - 设置指定便签为私密
  - 设置指定便签为提醒