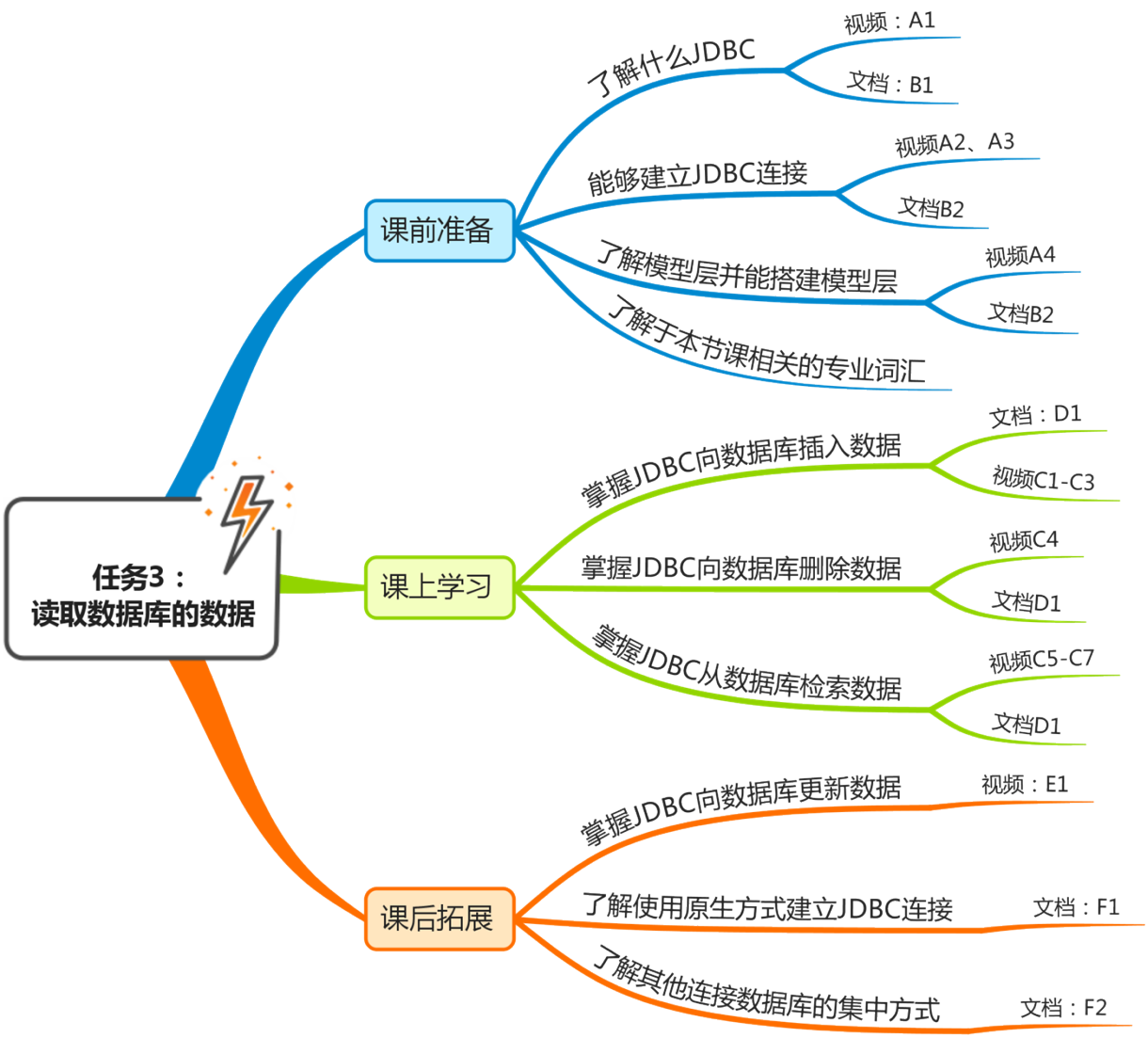
单元 1.1.3 用 JDBC 技术实现读取数据库

【课程导入】		
在本节课的学习中，我们将学习应用 JDBC 技术实现对 Mysql 数据库的连接、数据的查询和更改等操作，并对数据查询的分页技术进行详细的讲解。数据库操作是 JSP 技术的核心内容，必须扎实掌握本单元的知识。本单元为了便于学习，将数据库逻辑直接放置在 JSP 页面中，这样凡要进行数据操作的页面都要加上连接数据库的代码，不便于维护。		
知识目标	**能力目标**	**素质目标**
1. 掌握通过 JDBC 连接数据库并进行数据操作的方法，包括连接数据库、插入数据、读取数据、改写数据等 2.熟练掌握 Statement 对象的常用方法 3.熟练掌握 ResultSet 结果集打包和解包	1. 能够使用 JDBC 连接和操作数据库，包括连接数据库、插入数据、删除数据、读取数据、显示数据等。通 2.能够结合上学期学习的 MySQL 的数据库操作技巧，通过 JDBC 与数据库对话。	1.培养学生的团队意识和团队协作精神，锻炼学生的沟通交流能力; 2.通过项目教学，让学生真切的体验项目分析、设计、管理及实施的全过程;
学习任务	**重点难点**	**突破方法**
1. 能够通过 JDBC 对已有的数据表进行增删改查操作 2.能在 JSP 页面中调用 JDBC 显示数据库的数据	1. 连接数据库和读取数据库、写入数据库 2.JavaBean 与数据库记录之间的转化	采用翻转课堂、项目导入的教学模式，进行分组讨论、演示动画原理。

学习导航

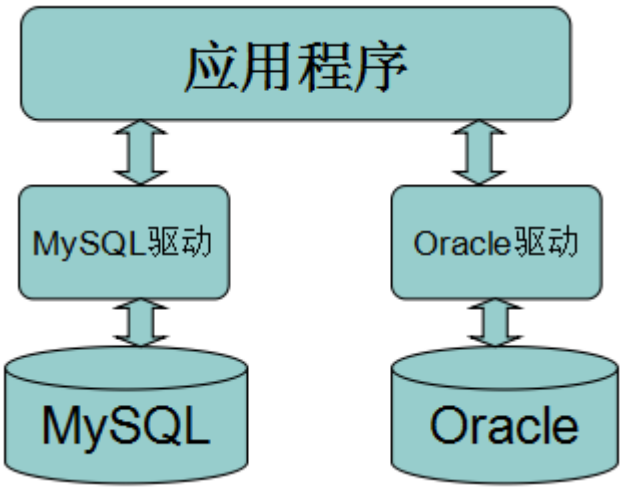


任务 1

了解 JDBC 的基本概念

1、数据库驱动

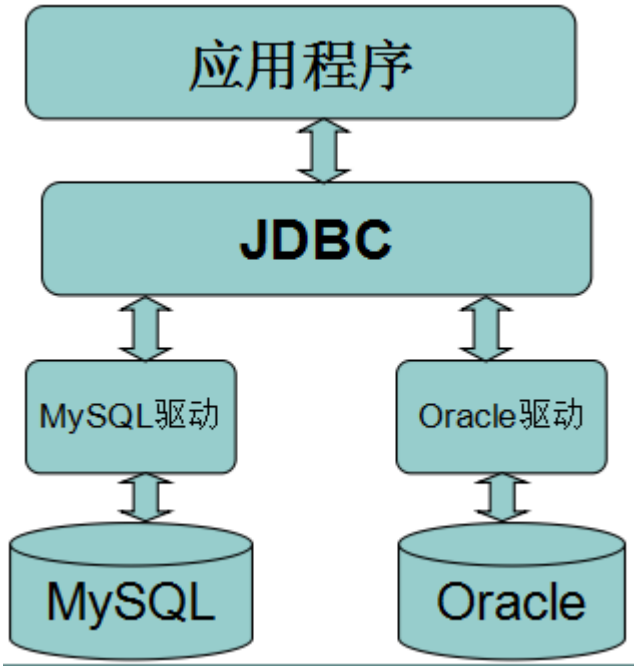
这里的驱动的概念和平时听到的那种驱动的概念是一样的，比如平时购买的声卡，网卡直接插到计算机上面是不能用的，必须要安装相应的驱动程序之后才能够使用声卡和网卡，同样道理，我们安装好数据库之后，我们的应用程序也是不能直接使用数据库的，必须要通过相应的数据库驱动程序，通过驱动程序去和数据库打交道，如下所示：



2、JDBC 介绍

SUN 公司为了简化、统一对数据库的操作，定义了一套 Java 操作数据库的规范（接口），称之为 JDBC。这套接口由数据库厂商去实现，这样，开发人员只需要学习 jdbc 接口，并通过 jdbc 加载具体的驱动，就可以操作数据库。

如下图所示：



JDBC 全称为：Java Data Base Connectivity（java 数据库连接），它主要由接口组成。
组成 JDBC 的 2 个包：

java.sql
javax.sql

开发 JDBC 应用需要以上 2 个包的支持外，还需要导入相应 JDBC 的数据库实现(即数据库驱动)。

任务 2

搭建 JDBC 环境

1、在 mysql 中创建一个库，并创建 user 表和插入表的数据。

SQL 脚本如下：

```
1 create database jdbcStudy character set utf8 collate utf8_general_ci;
2
3 use jdbcStudy;
4
5 create table users(
6     id int primary key,
7     name varchar(40),
8     password varchar(40),
9     email varchar(60),
10    birthday date
11 );
12
13 insert into users(id,name,password,email,birthday)
values(1,'zhansan','123456','zs@sina.com','1980-12-04');
14 insert into users(id,name,password,email,birthday)
values(2,'lisi','123456','lisi@sina.com','1981-12-04');
15 insert into users(id,name,password,email,birthday)
values(3,'wangwu','123456','wangwu@sina.com','1979-12-04');
```

2、新建一个 Java 工程，并导入数据驱动。

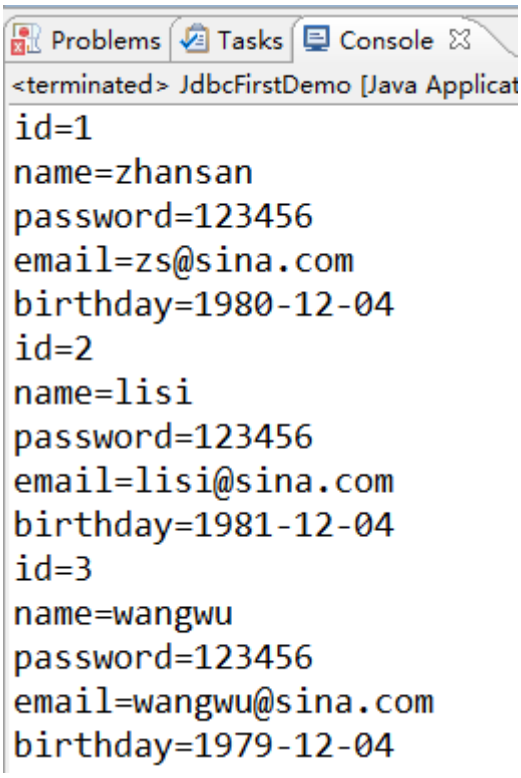
3、编写程序从 user 表中读取数据，并打印在命令行窗口中。

具体代码如下：

```
1 package me.gacl.demo;
2 import java.sql.Connection;
3 import java.sql.DriverManager;
4 import java.sql.ResultSet;
5 import java.sql.Statement;
6
```

```
 7 public class JdbcFirstDemo {
 8
 9     public static void main(String[] args) throws Exception {
10         //要连接的数据库 URL
11         String url = "jdbc:mysql://localhost:3306/jdbcStudy";
12         //连接的数据库时使用的用户名
13         String username = "root";
14         //连接的数据库时使用的密码
15         String password = "XDP";
16
17         //1.加载驱动
18         //DriverManager.registerDriver(new com.mysql.jdbc.Driver());不推荐
使用这种方式来加载驱动
19         Class.forName("com.mysql.jdbc.Driver");//推荐使用这种方式来加载驱
动
20         //2.获取与数据库的链接
21         Connection conn = DriverManager.getConnection(url, username,
password);
22
23         //3.获取用于向数据库发送 sql 语句的 statement
24         Statement st = conn.createStatement();
25
26         String sql = "select id,name,password,email,birthday from users";
27         //4.向数据库发 sql,并获取代表结果集的 resultset
28         ResultSet rs = st.executeQuery(sql);
29
30         //5.取出结果集的数据
31         while(rs.next()){
32             System.out.println("id=" + rs.getObject("id"));
33             System.out.println("name=" + rs.getObject("name"));
34             System.out.println("password=" + rs.getObject("password"));
35             System.out.println("email=" + rs.getObject("email"));
36             System.out.println("birthday=" + rs.getObject("birthday"));
37         }
38
39         //6.关闭链接，释放资源
40         rs.close();
41         st.close();
42         conn.close();
43     }
44 }
```

运行结果如下：



任务 3

实现对表的 GUID 操作

1.DriverManager 类讲解

Jdbc 程序中的 DriverManager 用于加载驱动，并创建与数据库的链接，这个 API 的常用方法：

1. DriverManager.registerDriver(new Driver())
2. DriverManager.getConnection(url, user, password)，

注意：在实际开发中并不推荐采用 **registerDriver** 方法注册驱动。原因有二：

1、查看 Driver 的源代码可以看到，如果采用此种方式，会导致驱动程序注册两次，也就是在内存中会有两个 Driver 对象。

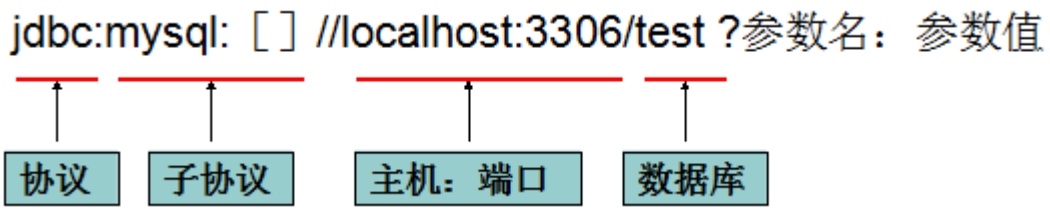
2、程序依赖 mysql 的 api，脱离 mysql 的 jar 包，程序将无法编译，将来程序切换底层数据库将会非常麻烦。

推荐方式：**Class.forName("com.mysql.jdbc.Driver");**

采用此种方式不会导致驱动对象在内存中重复出现，并且采用此种方式，程序仅仅只需要一个字符串，不需要依赖具体的驱动，使程序的灵活性更高。

2.数据库 URL 讲解

URL 用于标识数据库的位置，通过 URL 地址告诉 JDBC 程序连接哪个数据库，URL 的写法为：



常用数据库 URL 地址的写法：

- Oracle 写法：jdbc:oracle:thin:@localhost:1521:sid
- SqlServer 写法：jdbc:microsoft:sqlserver://localhost:1433; DatabaseName=sid
- MySql 写法：jdbc:mysql://localhost:3306/sid

如果连接的是本地的 Mysql 数据库，并且连接使用的端口是 3306，那么的 url 地址可以简写为：jdbc:mysql:///数据库

3.、Connection 类讲解

Jdbc 程序中的 Connection，它用于代表数据库的链接，Collection 是数据库编程中最重要的一个对象，客户端与数据库所有交互都是通过 connection 对象完成的，这个对象的常用方法：

- createStatement()：创建向数据库发送 sql 的 statement 对象。
- prepareStatement(sql) ：创建向数据库发送预编译 sql 的 PrepareSatement 对象。
- prepareCall(sql)：创建执行存储过程的 callableStatement 对象。
- setAutoCommit(boolean autoCommit)：设置事务是否自动提交。
- commit() ：在链接上提交事务。
- rollback() ：在此链接上回滚事务。

4.Statement 类讲解

Jdbc 程序中的 Statement 对象用于向数据库发送 SQL 语句， Statement 对象常用方法：

- executeQuery(String sql) ：用于向数据发送查询语句。
- executeUpdate(String sql)：用于向数据库发送 insert、update 或 delete 语句
- execute(String sql)：用于向数据库发送任意 sql 语句
- addBatch(String sql) ：把多条 sql 语句放到一个批处理中。
- executeBatch()：向数据库发送一批 sql 语句执行。

5.ResultSet 类讲解

Jdbc 程序中的 ResultSet 用于代表 Sql 语句的执行结果。Resultset 封装执行结果时，采用的类似于表格的方式。ResultSet 对象维护了一个指向表格数据行的游标，初始的时候，游标在第一行之前，调用 ResultSet.next() 方法，可以使游标指向具体的数据行，进行调用方法获取该行的数据。

ResultSet 既然用于封装执行结果的，所以该对象提供的都是用于获取数据的 get 方法：

获取任意类型的数据

getObject(int index)

getObject(string columnName)

获取指定类型的数据，例如：

getString(int index)
getString(String columnName)

ResultSet 还提供了对结果集进行滚动的方法：

- next()：移动到下一行
- Previous()：移动到前一行
- absolute(int row)：移动到指定行
- beforeFirst()：移动 resultSet 的最前面。
- afterLast() ：移动到 resultSet 的最后面。

6.释放资源

Jdbc 程序运行完后，切记要释放程序在运行过程中，创建的那些与数据库进行交互的对象，这些对象通常是 ResultSet, Statement 和 Connection 对象，特别是 Connection 对象，它是非常稀有的资源，用完后必须马上释放，如果 Connection 不能及时、正确的关闭，极易导致系统宕机。Connection 的使用原则是尽量晚创建，尽量早的释放。

为确保资源释放代码能运行，资源释放代码也一定要放在 finally 语句中。

任务 4

JDBC 实现数据批处理

1、使用 PreparedStatement 完成批处理范例

测试代码如下：

```
1 package me.gacl.demo;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import me.gacl.utils.JdbcUtils;
7 import org.junit.Test;
8
9 /**
10 * @ClassName: JdbcBatchHandleByStatement
11 * @Description: 使用 prepareStatement 实现 JDBC 批处理操作
12 * @author: 孤傲苍狼
13 * @date: 2014-9-20 下午 10:05:45
14 *
15 */
16 public class JdbcBatchHandleByPrepareStatement {
17
18     @Test
19     public void testJdbcBatchHandleByPrepareStatement(){
20         long starttime = System.currentTimeMillis();
21         Connection conn = null;
22         PreparedStatement st = null;
23         ResultSet rs = null;
```

```
24
25          try{
26              conn = JdbcUtils.getConnection();
27              String sql = "insert into testbatch(id,name) values(?,?)";
28              st = conn.prepareStatement(sql);
29              for(int i=1;i<1000008;i++){  //i=1000  2000
30                  st.setInt(1, i);
31                  st.setString(2, "aa" + i);
32                  st.addBatch();
33                  if(i%1000==0){
34                      st.executeBatch();
35                      st.clearBatch();
36                  }
37              }
38              st.executeBatch();
39          }catch (Exception e) {
40              e.printStackTrace();
41          }finally{
42              JdbcUtils.release(conn, st, rs);
43          }
44          long endtime = System.currentTimeMillis();
45          System.out.println("程序花费时间：" + (endtime-starttime)/1000 + "
秒！！");
46      }
47 }
```

2、采用 PreparedStatement.addBatch()方式实现批处理的优缺点

采用 PreparedStatement.addBatch()实现批处理

优点：发送的是预编译后的 SQL 语句，执行效率高。

缺点：只能应用在 SQL 语句相同，但参数不同的批处理中。因此此种形式的批处理经常用于在同一个表中批量插入数据，或批量更新表的数据。

关于 JDBC 批处理的内容就总结这么多。