数据库访问技术

本章知识点

- JDBC技术
  - JDBC的体系结构
  - JDBC驱动程序类型
- SQL语言
- MySQL数据库
  - MySQL的安装、配置
  - 在MySQL环境下创建数据库及数据表

JDBC中的主要接口和类的使用

DriverManager类

Connection接口

Statement接口

PreparedStatement接口

ResultSet接口

数据库开发模式

关于JDBC

- JDBC（Java Database Connectivity）技术：以统一的方式、非常方便地操作各种主流数据库。
- JDBC API：Sun制定的操作数据库的标准，与数据库操作相关的部分都是接口，没有提供实现类。这些实现类由每个数据库厂商依据接口标准提供，实现类即相当于该数据库的驱动程序。
- JDBC开发的数据库应用既可以跨平台，也可以跨数据库运行：使用标准的SQL。

13.1 MySQL数据库与SQL语法

- MySQL是一个多用户、多线程的数据库，由瑞典MySQL AB公司开发，2010年被甲骨文公司收购。目前MySQL被广泛地应用在Internet上的中小型网站中。其特点是体积小、速度快、总体拥有成本低，为开源软件， MySQL已经成为目前最受欢迎的中小型企业数据库之一。

- MySQL官方网址：www.mysql.com

13.1.1 MySQL数据库的安装和配置

系统变量	说明	安装时指定gbk后的my.ini配置文件
character__set_client	从客户端发送给服务器的语句的字符集	[mysql] default-character-set=gbk
character_set_connection	客户端和服务器连接的字符集	
character_set_results	从服务器发送到客户端的select语句的最终结果的字符集	
character_set_database	当前数据库的默认字符集	[mysqld] character-set-server=gbk
character_set_server	服务器的默认字符集	

13.1.2 MySQL数据库的基本命令

1. 数据库操作命令

功能	命令
创建数据库	create database [if not exists] 数据库名 [default character set …];
显示数据库	show databases;
连接数据库	use 数据库名;
删除数据库	drop 数据库名;

13.1.2 MySQL数据库的基本命令

2. 数据表操作命令

功能	命令
建表	create table <表名> (<字段名 1> <类型 1> [,..<字段名 n> <类型 n>]) [character set=…];
显示表	show tables;
查看表结构	desc 表名;
删除表	drop table <表名>;
更改表名	rename table 原表名 to 新表名;
在表中增加字段	alter table 表名 add 字段名 类型 [default …];
在表中删除字段	alter table 表名 drop column 字段名;
修改字段名称/类型	alter table 表名 change 原字段名 新字段名 新类型;

13.1.3 SQL语句

- 在使用Java语言开发关于数据库的应用程序中主要使用insert、 delete、update、select 4个命令(增删改查)。

功能	命令
插入记录	insert into <表名> [(<字段名 1>[,<字段名2>…])] values(值1)[(值2)…]
更新记录	update <表名> set 字段名1=值1[,字段名2=值2]…[where 条件]
删除记录	delete from <表名> [where 条件]

13.1.3 SQL语句

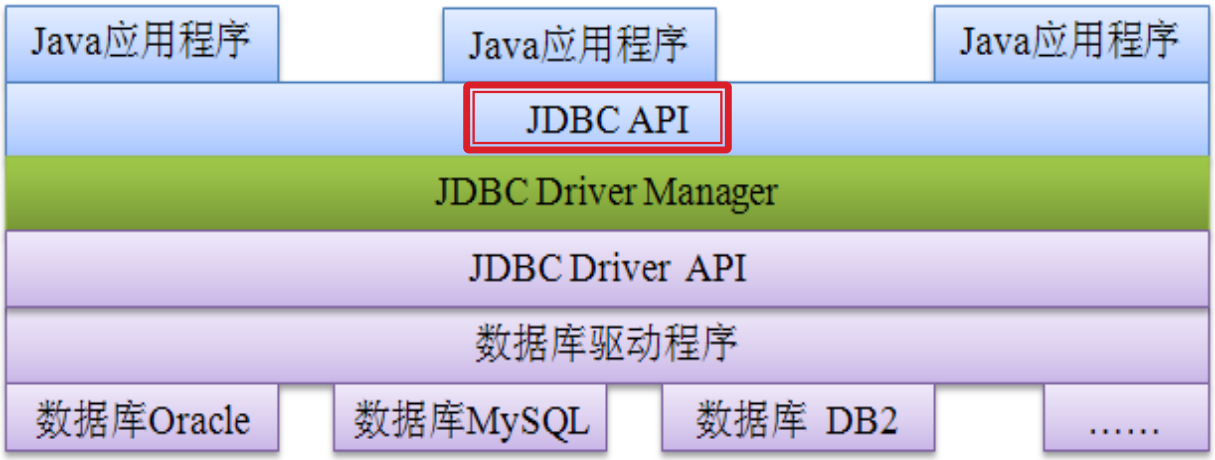
- 在使用Java语言开发关于数据库的应用程序中主要使用insert、 delete、update、select 4个命令(增删改查)。

功能	命令
基本型	select * from <表名> [where 条件] select [字段1,字段2,…] from <表名> [where 条件]
清除重复记录	select [distinct] ……
对查询结果排序	select … [order by 字段1[desc],…]

13.2 JDBC的体系结构和JDBC驱动程序的实现方式

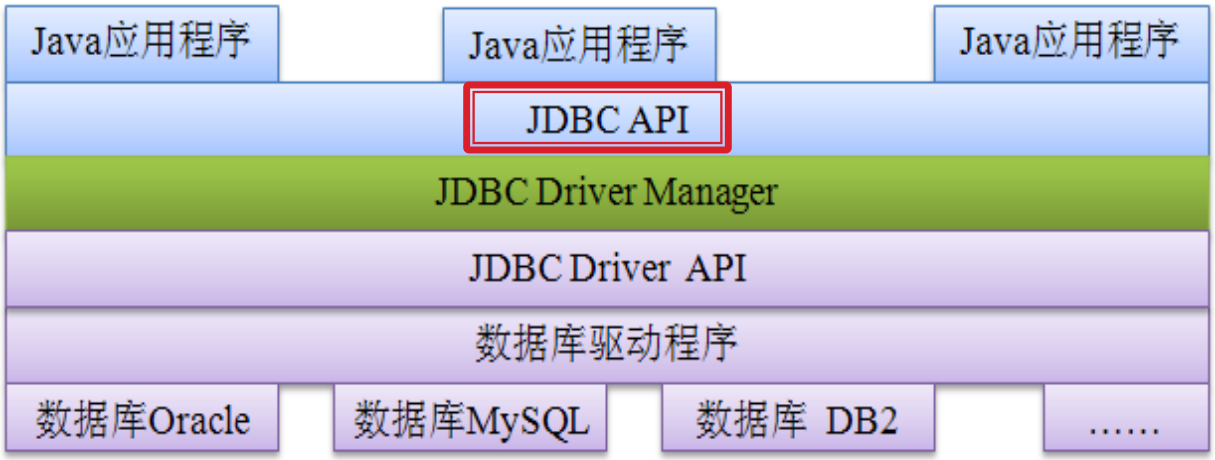
13.2.1 JDBC的体系结构

- JDBC API是为Java程序员提供的，其作用是屏蔽不同的数据库驱动程序之间的差别，使Java程序员有一个标准的、纯Java的数据库程序设计接口，使Java可以访问任意类型的数据库。



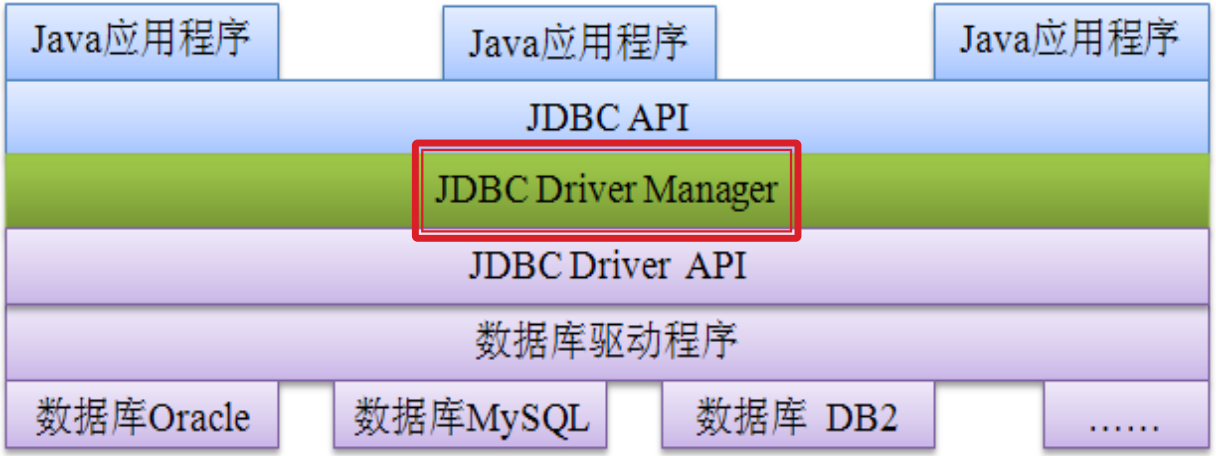
13.2.1 JDBC的体系结构

- JDBC API是为Java程序员提供的，其作用是屏蔽不同的数据库驱动程序之间的差别，使Java程序员有一个标准的、纯Java的数据库程序设计接口，使Java可以访问任意类型的数据库。

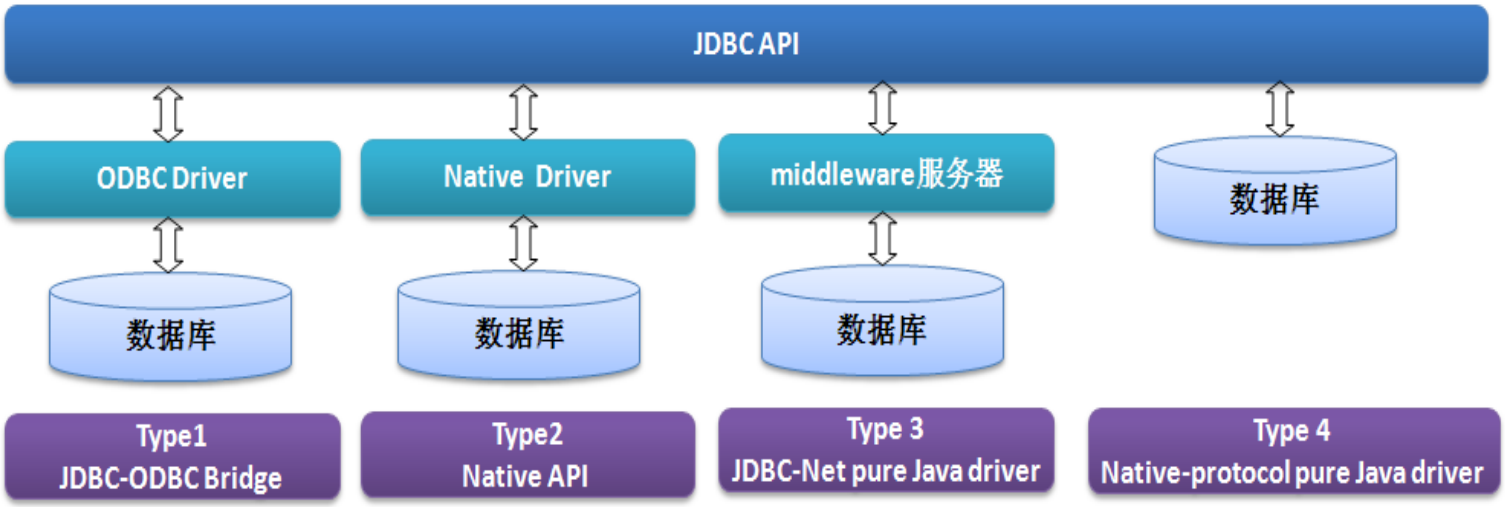


13.2.1 JDBC的体系结构

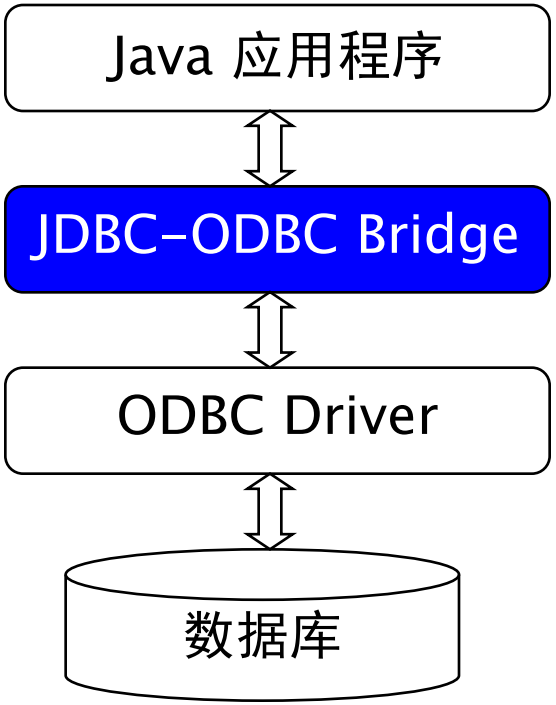
- JDBC Driver Manager工作在Java应用程序与数据库驱动程序之间，为应用程序加载和调用驱动程序。Java应用程序首先使用JDBC API来与JDBC Driver Manager交互，由JDBC Driver Manager载入指定的数据库驱动程序，之后就可以由JDBC API直接存取数据库。



13.2.2 JDBC驱动程序的实现方式

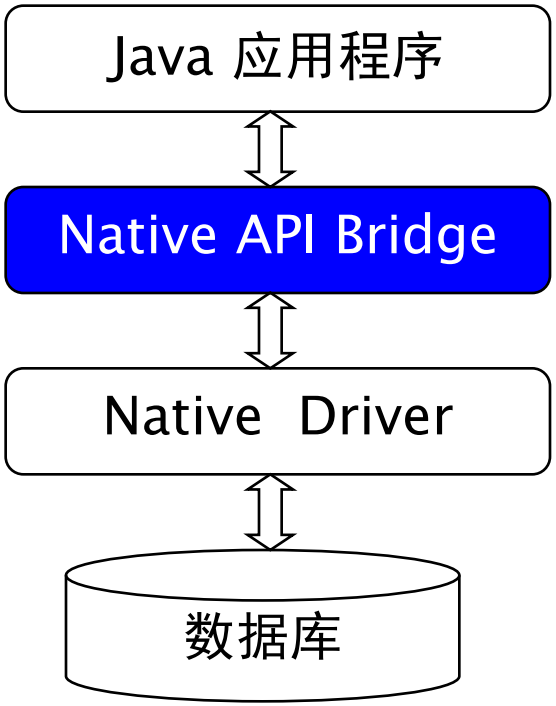


13.2.2 JDBC驱动程序的实现方式



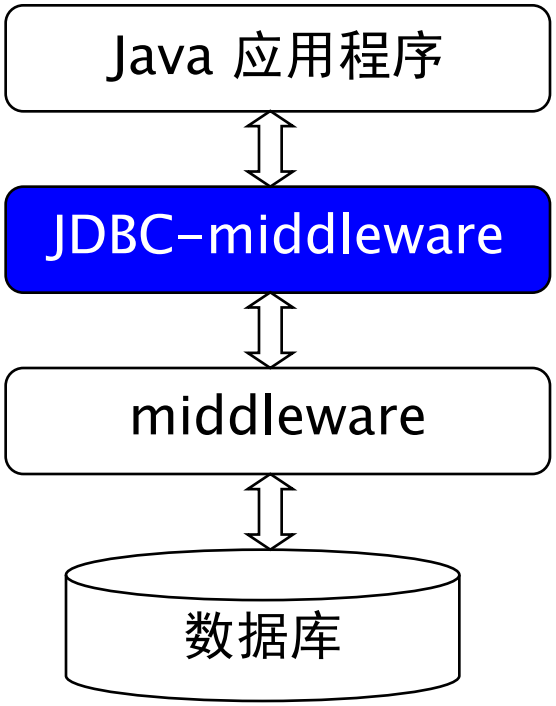
- Type 1：**JDBC-ODBC bridge plus ODBC driver**：JDBC-ODBC桥接驱动程序。其底层通过ODBC(Open Database Connectivity)驱动程序来连接数据库。

13.2.2 JDBC驱动程序的实现方式



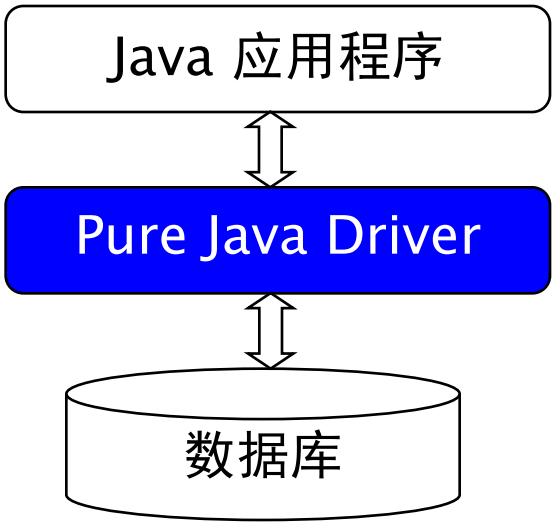
- Type 2：**Native-API partly-Java driver**：本地 API-部分用Java来编写的驱动程序。

13.2.2 JDBC驱动程序的实现方式



- Type 3：**JDBC-Net pure Java driver**：JDBC 网络纯 Java 驱动程序。

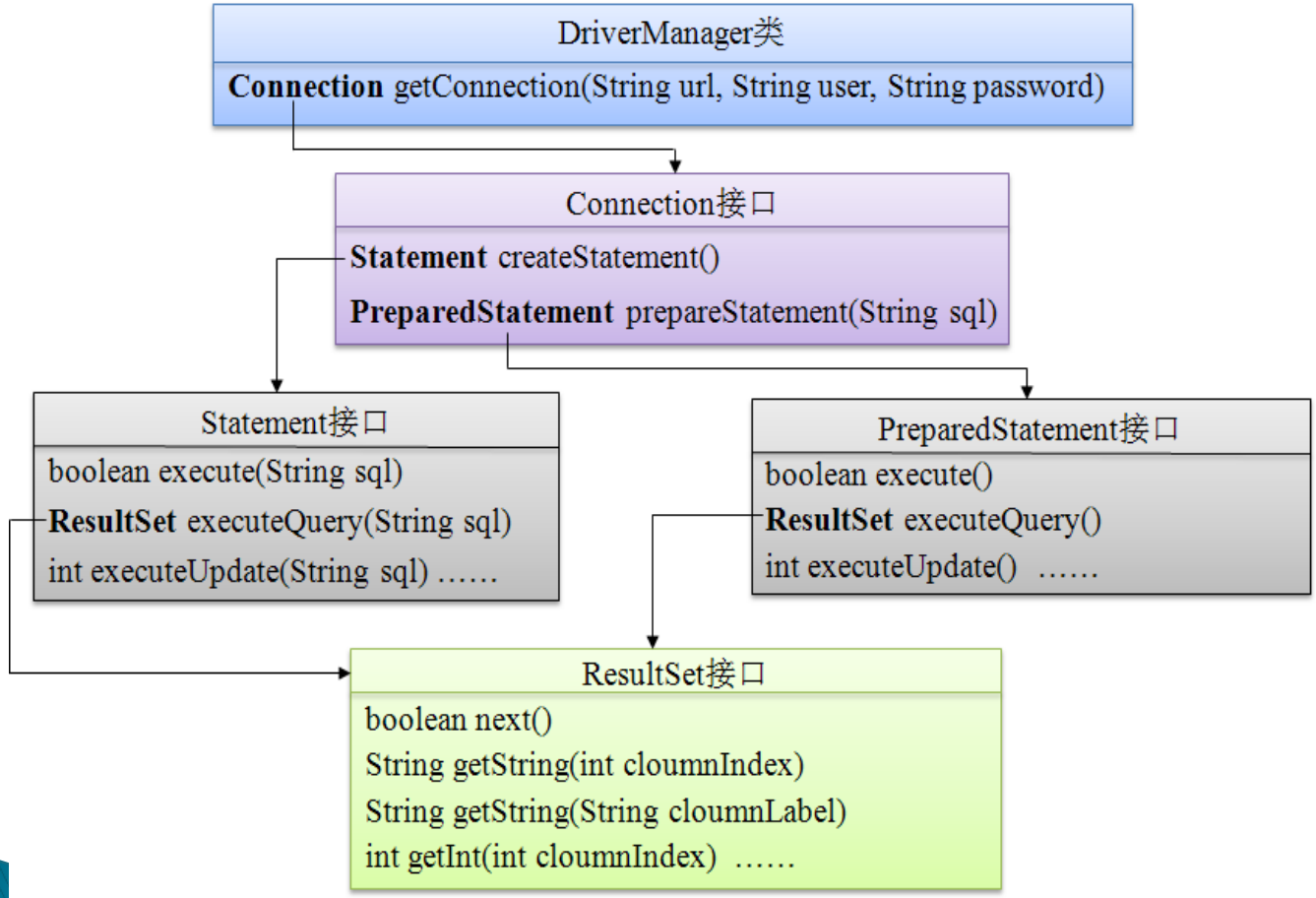
13.2.2 JDBC驱动程序的实现方式



- Type 4：**Native-protocol pure Java driver**：本地协议纯 Java驱动程序。

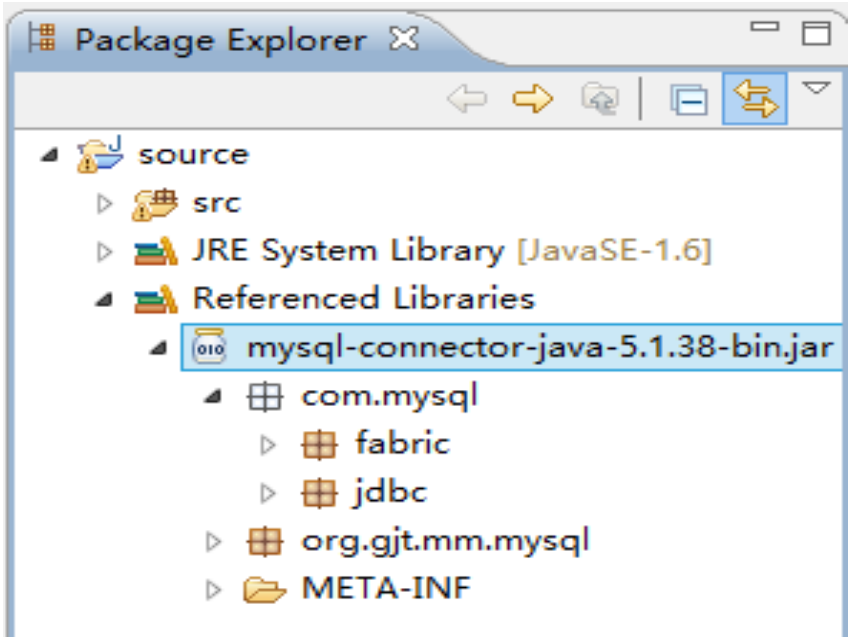
13.3 建立JDBC数据库连接

13.3.1 JDBC API的主要类和接口



13.3.2 连接数据库

1．导入数据库驱动程序jar文件



13.3.2 连接数据库

3．建立数据库连接

```
Connection con =
DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/
test ", "root", "1234");
```

Connnetion getConnection(String url, String user, String password)

URL	访问数据库的路径	jdbc:	subProtocol:	subName
		主通信协议	子通信协议	主机地址+数据库文件名称
user	数据库用户名			
password	用户密码			

13.3.2 连接数据库

4．自定义数据库连接工具类

（1）配置文件

- 为了提高Java应用程序的可移植性，通常情况下，数据库连接的驱动、URL字符串、用户名和密码不会直接写在程序中，而是通过读取配置文件导入。这样，无论Java程序需要连接哪一种数据库，都只需将它们的连接信息写入配置文件即可。

配置文件database.properties：

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql://127.0.0.1:3306/test
user=root
password=1234
```

13.4 使用JDBC访问数据库

使用JDBC编写数据库应用程序的步骤通常包括：

- （1）加载驱动
- （2）建立数据库连接
- （3）创建Statement或PreparedStatement对象
- （4）执行SQL语句
- （5）如果有ResultSet结果集，则对其进行处理
- （6）释放资源

13.4.1 Statement与数据表的增、删、改

【例13-2】封装2个方法，完成向数据表user中添加记录和修改记录的操作。

```
"values('grace@126.com','grace','学习') "
"values('"+user.getEmail()+"','"+user.getUsername()+
"','"+user.getHobbies()+"')";
```

字符串拼接的工程非常浩大、繁琐、易错。
对于需要传入参数的SQL语句的处理，强烈建议使用PreparedStatement对象处理。

13.4.2 PreparedStatement与数据表的增、删、改

- PreparedStatement的两个优势

（1）它用预编译的方式包装SQL语句，数据库不必每次编译SQL语句，因此性能比Statement好。

```
insert into user values(null, 'lucy@126.com','lucy','体育运动');
insert into user values(null, 'leo@126.com', 'leo','看书');
```

```
insert into user values(null,?,?,?);
```

在执行SQL语句前向占位符“？”传递取值!

13.4.3 数据表的查询与ResultSet

1．ResultSet的类型

Statement createStatement(int resultSetType, int resultSetConcurrency)

PreparedStatement prepareStatement(String sql, int resultSetType, int resultSetConcurrency)

CONCUR_READ_ONLY（只读）
CONCUR_UPDATABLE（可修改）

TYPE_FORWARD_ONLY(只允许向前访问一次)
TYPE_SCROLL_INSENSITIVE(向前或向后移动)
TYPE_SCROLL_SENSITIVE(向前或向后移动+会受到其他用户对数据库所做更改的影响)

13.4.3 数据表的查询与ResultSet

2．ResultSet的迭代—行操作

- 获取到ResultSet结果集后，初始的指针指在结果集第一条记录的前面。
- 如果可以确定select查询的结果只包含一条记录，则使用**if**(rs.next()){……}控制迭代；
- 如果select查询的结果集包含多条记录，使用**while**(rs.next()){……}控制迭代。

3．ResultSet的getXxx()方法—列操作

13.4.3 数据表的查询与ResultSet

【例13-4】向user表中增加一个标识最后一次登录日期的字段，插入几条记录后查询、打印某个指定日期之后的所有记录。

Java中在java.util包和java.sql包下各自有一个Date类，java.sql包下的Date类用于从数据表读出的date类型数据的存储，java.util包下的Date类则用于与之配合的客户端程序。所以在实体类中，lastLoginDate成员的数据类型应为java.util.Date。

13.5 综合实践--数据库访问的开发模式

带有数据库的应用系统，JDBC访问的架构分为以下几部分。

（1）实体类

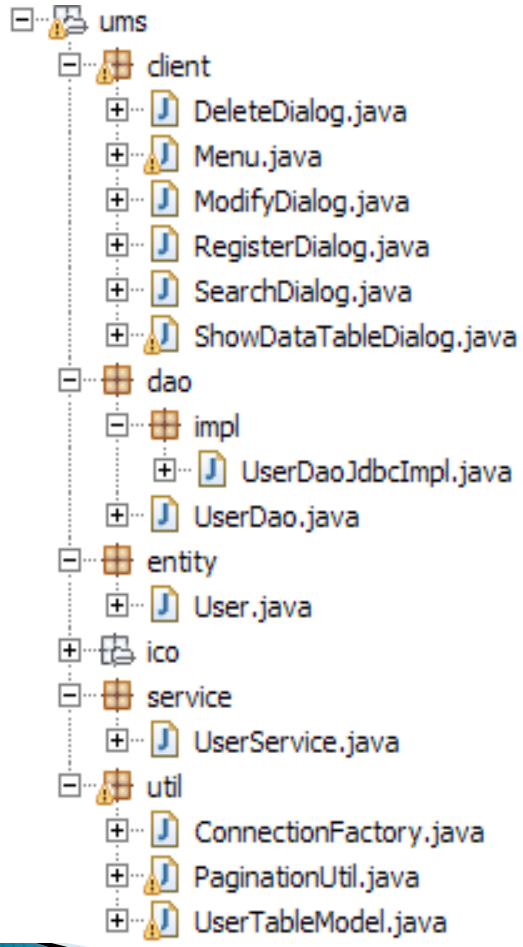
（2）DAO接口

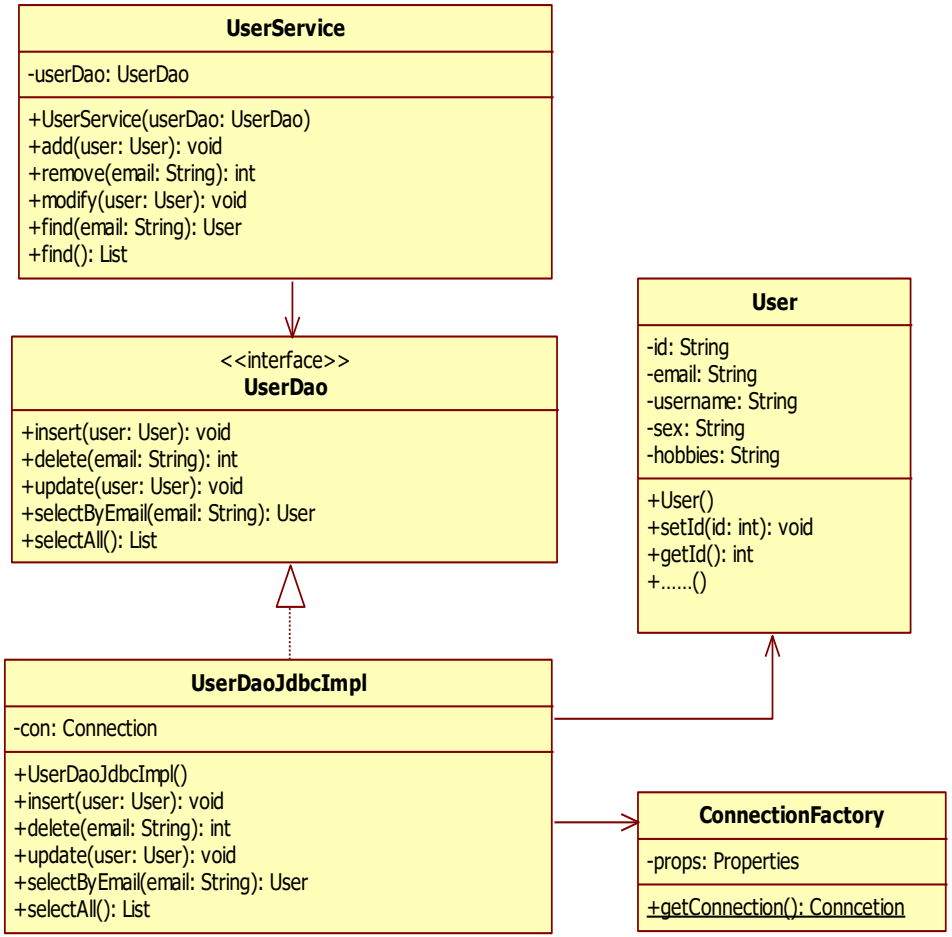
（3）DAO实现类

（4）业务层类

（5）应用程序类

13.5.1 基于数据库存储的用户管理系统





13.5.2 业务层--封装DAO中的方法

- 业务层负责封装DAO层的方法，用户管理系统中的业务逻辑比较简单，所以就是直接调用DAO层的方法。注意：业务层方法将增、删、改、查操作从业务层面进行命名，分别为add()、remove()、modify()、find()。

13.5.3 应用层—调用业务层方法完成系统功能

- 各个对话框类在事件处理时使用UserService对象完成相应业务功能，UserService对象中封装的UserDao对象按需创建即可。

本章思维导图


数据库访问技术
I. 数据库基础知识
MySQL
客户端/服务器体系结构
安装要点
端口
3306
字符集
客户端: set names gbk
建库、建表指定utf8
常用命令
库
create database.....
use 数据库名;
.....
表
create table .....
show tables;
desc 表名;
.....
字段
not null
primary key/primary key()
unique
SQL语句
insert
delete
update
select
II. JDBC
体系结构
JDBC API
Java程序员使用
数据库应用程序
JDBC Driver Manager
JDBC Driver API
数据库厂商使用
数据库驱动程序
驱动方式
Type1
JDBC-ODBC桥
实验性题目
Type2
本地API
非web应用程序
Type3
JDBC网络驱动
web应用程序
连接多种类型数据库的应用
Type4
纯Java驱动
web应用程序
连接单一数据库的应用
III. 数据库访问过程
加载驱动
类加载器
Class.forName(驱动类)
实例驱动
创建驱动类实例对象
DriverManager.registerDriver(对象)
连接数据库
DriverManager
getConnection()
参数
读取配置文件
Properties
URL
主通信协议
子通信协议
主机地址
数据库文件名
user
password
返回值
Connection对象
创建Statement/PreparedStatement对象
Statement
数据库连接.createStatement()
PreparedStatement
预编译SQL语句
数据库连接.createPreparedStatement(sql语句)
执行SQL语句
增删改
executeUpdate()
查询
select
ResultSet结果集
处理select查询结果
迭代
行
结果集对象.next()
访问
列
结果集对象.getXxx()
释放数据库资源
finally()
IV. 开发模式
entity包
对应每张数据表
实体类
dao包
数据表访问原子操作
XxxDao接口
XxxDao接口实现类
dao.impl包
service包
调用Dao方法组成业务逻辑
业务层
client包
应用层
V. 案例
【例13-1】建立用户信息表
create table
【例13-2】向数据表添加和修改记录
数据库访问
Statement
增删改
【例13-3】使用PreparedStatement对象完成添加记录
PreparedStatement
【例13-4】按日期查询数据表
查询
Date型字段的处理
ResultSet结果集
【综合实践】数据库访问开发模式
数据库应用开发模式
删除用户
用户管理系统
【探究与实践】用户管理系统的权限管理
数据库应用开发模式