JDBC数据库操作

讲师：高淇

JDBC课程规划

- 数据库知识：
  - Mysql数据库的安装、配置、使用
  - navicat客户端软件的使用、命令行操作
- JDBC技术
  - 架构介绍、Driver、DriverManager、Connection
  - Statement　　Sql注入
  - preparedStatement　基本用法　参数处理技巧
  - ResultSet　　结果集处理
  - 时间类型　　CLOB　　BLOB
  - CallableStatement
  - 批处理
  - 事务管理　加载大sql文件技巧
  - 经典JDBC代码总结
  - 使用properties文件存储JDBC连接信息

JDBC课程规划

- JDBC工具类
  - Apache commons的DbUtils
- 数据库连接池
  - C3P0
  - DBCP
  - Proxool
- 项目：
  - 结合设计模式写出自己的连接池
  - 测试连接池的效率
- ORM基本思想
- 项目：
  - 结合设计模式写出自己的ORM框架，并增加缓存设计

Mysql数据库简介

- Mysql特点
  - 是一种开放源代码的关系型数据库管理系统（RDBMS）
  - 目前很多大公司(新浪、京东、阿里等)都在使用mysql
  - 适应于所有的平台
  - 支持多线程，充分利用CPU资源，性能很出色
  - 价格便宜
  - 大数据库处理。
    - 对某些包含 50,000,000 个记录的数据库使用MySQL完全没有问题
  - 使用最多的版本是5.5.

Mysql数据库的安装

- Mysql的下载
  - 官方主页：http://www.mysql.com/
- Mysql的安装

- 启动和停止Mysql服务

Navicat客户端软件的使用

- 操作mysql数据库的利器：navicat
- Navicat安装
- Navicat的基本使用方式
  - 建数据库
  - 建表
  - 查询
  - 导入sql数据
  - 导出查询结果

Mysql数据库的命令行操作

- 配置环境变量
  - 将bin目录配置到path中
- 命令行操作

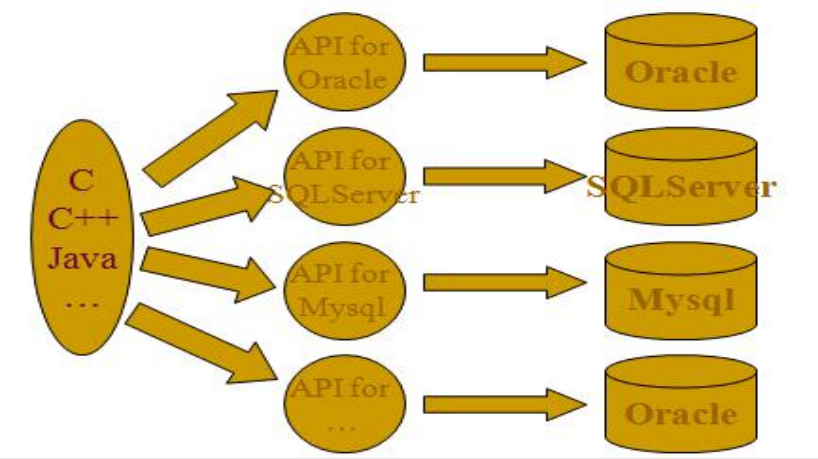
登陆操作	**mysql -hlocalhost –uroot –p123456**
退出操作	exit
数据库操作	建库：create database 库名; 卸载库：drop database 库名; 显示所有数据库：show databases; 选择库：use testjdbc;
表操作	建表的sql语句； 显示库中所有表：show tables; 显示某个表的结构：describe testjdbc;
SQL操作	select语句；Insert语句； update语句； delete语句； 表操作DDL语句(create, alter, drop等)；

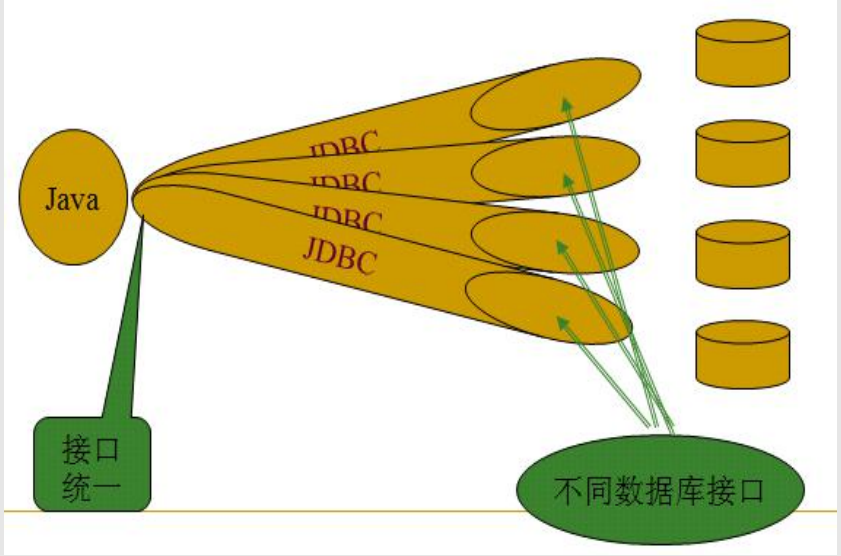
什么是JDBC?

- JDBC(Java Database Connection)为java开发者使用数据库提供了**统一的编程接口**，它由一组java类和接口组成。是java程序与数据库系统通信的标准API。JDBC API 使得开发人员可以使用纯java的方式来连接数据库，并执行操作。

- sun公司由于不知道各个主流商用数据库的程序代码，因此无法自己写代码连接各个数据库，因此，sun公司决定，自己提供一套api，凡是数据库想与Java进行连接的，数据库厂商自己必须**实现JDBC这套接口**。而数据库厂商的JDBC实现，我们就叫他此数据库的**数据库驱动**。

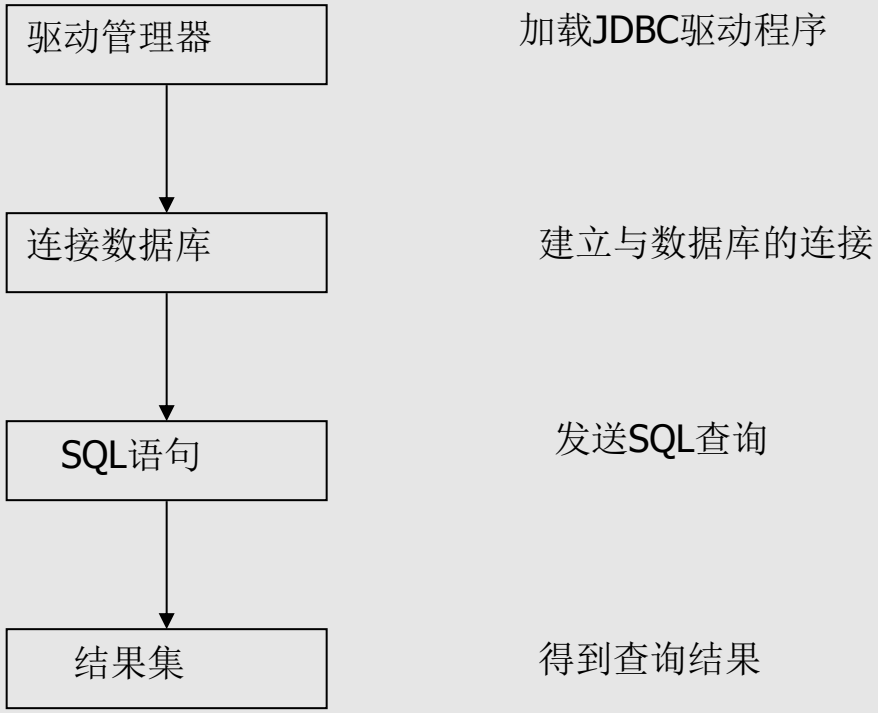
Why JDBC?





JDBC访问数据库流程

- 访问数据库流程：



JDBC常用接口

- Driver接口
  - Driver接口由数据库厂家提供，对于java开发者而言，只需要使用Driver接口就可以了。
  - 在编程中要连接数据库，必须先装载特定厂商的数据库驱动程序。不同的数据库有不同的装载方法。
  - 驱动：就是各个数据库厂商实现的Sun公司提出的JDBC接口。 即对Connection等接口的实现类的jar文件
  - 装载MySql驱动
    - Class.forName("com.mysql.jdbc.Driver");
  - 装载Oracle驱动
    - Class.forName("oracle.jdbc.driver.OracleDriver");

JDBC常用接口

- DriverManager接口
  - DriverManager是JDBC的管理层，作用于用户和驱动程序之间。
  - DriverManager跟踪可用的驱动程序，并在数据库和相应的驱动程序之间建立连接。

JDBC常用接口

- Connection**接口**
  - Connection与特定数据库的连接（会话）,在连接上下文中执行 SQL 语句并返回结果。
  - DriverManager的getConnection()方法建立在JDBC URL中定义的数据库Connection连接上
  - 连接MYSQL数据库：
    - Connection con = DriverManager.getConnection("jdbc:mysql://host:port/database","user","password");
  - 连接ORACLE数据库：
    - Connection con = DriverManager.getConnection("jdbc:oracle:thin:@host:port:databse","user","password");

JDBC常用接口

- Statement接口
  - 用于执行静态 SQL 语句并返回它所生成结果的对象。
  - 三种Statement类：
    - Statement：
      - 由createStatement创建，用于发送简单的SQL语句。(不带参数的)
    - PreparedStatement：
      - 继承自Statement接口，由prepareStatement创建，用于发送含有一个或多个输入参数的sql语句。PreparedStatement对象比Statement对象的**效率更高**，并且**可以防止SQL注入**。我们一般都用PreparedStatement.
    - CallableStatement：
      - 继承自PreparedStatement 。由方法preParecall创建，用于调用**存储过程**。
  - 常用的Statement方法：
    - execute()：运行语句，返回是否有结果集。
    - executeQuery()：运行select语句，返回ResultSet结果集。
    - executeUpdate()：运行insert/update/delete操作，返回更新的行数。

JDBC常用接口

- ResultSet接口
  - Statement执行SQL语句时返回ResultSet结果集。
  - ResultSet提供的检索不同类型字段的方法，常用的有：
    - getString():获得在数据库里是varchar、char等数据类型的对象。
    - getFloat():获得杂数据库里是Float类型的对象。
    - getDate():获得在数据库里面是Date类型的数据。
    - getBoolean():获得在数据库里面是Boolean类型的数据

- 依序关闭使用之对象及连接:
  - ResultSet →Statement→ Connection

JDBC详细操作

- 灵活指定SQL语句中的变量
  - PreparedStatement
- 对存储过程进行调用
  - CallableStatement
- 运用事务处理
  - Transaction
- 批处理
  - Batch
  - 对于大量的批处理，建议使用Statement，因为PreparedStatement的预编译空间有限，当数据量特别大时，会发生异常。

事务

- **事务基本概念**
  - **一组要么同时执行成功，要么同时执行失败的SQL语句。是数据库操作的一个执行单元！**
  - **事务开始于：**
    - 连接到数据库上，并执行一条DML语句(INSERT、UPDATE或DELETE)。
    - 前一个事务结束后，又输入了另外一条DML语句。
  - **事务结束于：**
    - 执行COMMIT或ROLLBACK语句。
    - 执行一条DDL语句，例如CREATE TABLE语句；在这种情况下，会自动执行COMMIT语句。
    - 执行一条DCL语句，例如GRANT语句；在这种情况下，会自动执行COMMIT语句。
    - 断开与数据库的连接。
    - 执行了一条DML语句，该语句却失败了；在这种情况中，会为这个无效的DML语句执行ROLLBACK语句。

事务

- 事务的四大特点（ACID）
  - atomicity（原子性）
    - 表示一个事务内的所有操作是一个整体，要 么全部成功，要么全失败；
  - consistency（一致性）
    - 表示一个事务内有一个操作失败时，所有的更改过的数据都必须回滚到修改前的状态；
  - isolation（隔离性）
    - 事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。
  - durability（持久性）
    - 持久性事务完成之后，它对于系统的影响是永久性的。

事务

- 事务隔离级别从低到高：
  - **读取未提交（Read Uncommitted)**
  - **读取已提交(Read Committed)**
  - 可重复读（Repeatable Read)
  - 序列化（serializable)

时间类型

- java.util.Date
  - 子类：java.sql.Date　　表示年月日
  - 子类：java.sql.Time　　表示时分秒
  - 子类：java.sql.Timestamp　　表示年月日时分秒

- 日期比较处理
  - 插入随机日期
  - 取出指定日期范围的记录

```
public static  long  str2Date(String dateStr){
  DateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
                try {
                                return format.parse(dateStr).getTime();
                } catch (ParseException e) {
                                e.printStackTrace();
                                return 0;
                }
}
```

```
ps = conn.prepareStatement("select * from t_user where lastLoginTime>? and
lastLoginTime<?  order by lastLoginTime ");
Timestamp start = new Timestamp(str2Date("2015-4-18 8:10:20"));
Timestamp end = new Timestamp(str2Date("2015-4-18  9:9:10"));
ps.setObject(1, start);
ps.setObject(2, end);
```

CLOB

- CLOB（Character Large Object）
  - 用于存储大量的文本数据
  - 大字段有些特殊，不同数据库处理的方式不一样，大字段的操作常常是**以流的方式**来处理的。而非一般的字段，一次即可读出数据。
- Mysql中相关类型：
  - TINYTEXT最大长度为**255**(2^[8]–1)字符的TEXT列。
  - TEXT[(M)]最大长度为**65,535**(2^[16]–1)字符的TEXT列。
  - MEDIUMTEXT最大长度为**16,777,215**(2^[24]–1)字符的TEXT列。
  - LONGTEXT最大长度为**4,294,967,295或4GB**(2^[32]–1)字符的TEXT列。

BLOB

- BLOB（Binary Large Object）
  - 用于存储大量的二进制数据
  - 大字段有些特殊，不同数据库处理的方式不一样，大字段的操作常常是**以流的方式来处理**的。而非一般的字段，一次即可读出数据。
- Mysql中相关类型：
  - TINYBLOB最大长度为**255**(2^[8]–1)字节的BLOB列。
  - BLOB[(M)]最大长度为**65,535**(2^[16]–1)字节的BLOB列。
  - MEDIUMBLOB最大长度为**16,777,215**(2^[24]–1)字节的BLOB列。
  - LONGBLOB最大长度为**4,294,967,295或4GB**(2^[32]–1)字节的BLOB列。

经典JDBC代码总结

- JDBC代码操作总结

- 使用资源文件存储数据库连接信息

```
mysqlDriver=com.mysql.jdbc.Driver
mysqlURL=jdbc\:mysql\://localhost\:3306/testjdbc
mysqlUser=root
mysqlPwd=123456

oracleDriver=oracle.jdbc.driver.OracleDriver
oracleURL=jdbc\:oracle\:thin\:@localhost\:1521\:orcl
oracleUser=scott
oraclePwd=tiger
```

ORM基本思想

- ORM(Object Relationship Mapping)的基本思想
  - **表结构跟类对应； 表中字段和类的属性对应；表中记录和对象对应；**
  - 让javabean的属性名和类型尽量和数据库保持一致！
  - 一条记录对应一个对象。将这些查询到的对象放到容器中(List,Set,Map)

- 将表中的一条记录封装到Object数组中
- 将表中的一条记录封装到map中
- 将表中一条记录封装到javabean对象中