第7讲 数据库技术

1. 数据库

- 数据库（Database）是按照数据结构来组织、存储和管理数据的仓库。
- 每个数据库都有一个或多个不同的 API 用于创建，访问，管理，搜索和复制所保存的数据。
- 当前的数据库都是关系型数据库，是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。

数据库表举例

id	bookname	author	price
1	射雕英雄传	金庸	54
2	红楼梦	曹雪芹	59
3	水浒传	施耐庵	30
4	西游记	吴承恩	94

一张名为books的数据库表，有4列，共4条记录，id列为主键

- 关系型数据库管理系统的特点：

1. 数据以表格的形式出现
2. 每行为一条记录
3. 每列为记录所对应的数据域
4. 许多的行和列组成一张表（table）
5. 若干互相关联的表（table）组成一个数据库

数据库相关概念

- 数据表: 表是数据的矩阵
- 列: 一列包含了相同类型的数据。
- 行：一行（=元组，或记录）是一组相关的数据
- 字段：每行中每列数据称为一个字段。
- 主键：主键是能够区分一行一行记录的列。

2. SQL语句

- SQL 是一门 ANSI 的标准计算机语言，用来访问和操作数据库系统。
- SQL主要功能：
  - 面向数据库执行查询
  - 从数据库取回数据
  - 在数据库中插入新的记录
  - 更新数据库中的数据
  - 从数据库删除记录

SQL举例

- 查询books表中的所有记录的所有字段

```
select id, bookname, author, price
from books;
```

- 查询books表中的id为1的书名和作者字段

```
select bookname, author
from books
where id=1;
```

SQL举例

- 插入一条新的记录

```
insert into books
        (id, bookname, author, price)
values
        (5,  ‘三国演义’,  ‘罗贯中’, 60);
```

SQL举例

- 将books表中id为1的记录的价格字段的值修改为为100

```
update books
set price = 100
where id = 1
```

- 删除books表中id为1的记录

```
delete from books
where id = 1
```

3. JDBC技术

- JDBC的全称为Java DataBase Connectivity，是一套面向对象的应用程序接口（API），制定了统一的访问各类关系数据库的标准接口，为各个数据库厂商提供了标准接口的实现。
- 通常情况下使用JDBC完成以下操作：

  （1）同数据库建立连接；

  （2）向数据库发送SQL语句；

  （3）处理从数据库返回的结果。

4. JDBC常用类

- 驱动程序Driver
- 驱动程序管理器DriverManager
- 数据库连接Connection
- 语句Statement
- 语句PreparedStatement
- 访问结果集ResultSet

a. 驱动程序Driver

- 通过java.lang.Class类的静态方法forName(String className)，加载要连接数据库的Driver类，该方法的入口参数为要加载Driver类的完整包名。

b. 驱动程序管理器DriverManager

- DriverManager类负责管理JDBC驱动程序的基本服务，是JDBC的管理层，作用于用户和驱动程序之间，负责跟踪可用的驱动程序，并在数据库和驱动程序之间建立连接。
- getConnection(String url, String user, String password)：静态方法，用来获得数据库连接，有3个入口参数，依次为要连接数据库的URL、用户名user和密码password，返回值类型为一个Connection对象

c. 数据库连接Connection

- Connection接口负责与特定数据库的连接，在连接的的上下文中可以执行SQL语句并返回结果。
- createStatement()方法：创建并返回一个Statement实例，通常在执行无参数的SQL语句时创建该实例

- prepareStatement()：创建并返回一个PreparedStatement实例，通常在执行包含参数的SQL语句时创建该实例，并对SQL语句进行了预编译处理
- close()方法：立即释放Connection实例占用的数据库和JDBC资源，即关闭数据库连接

d. 语句Statement

- Statement用来执行静态的SQL语句，并返回执行结果
- executeQuery(String sql)方法：执行指定的静态SELECT语句，并返回一个永远不能为null的ResultSet实例
- executeUpdate(String sql)方法：执行指定的静态INSERT、UPDATE或DELETE语句，并返回一个int型数值，为同步更新记录的条数
- close()方法：立即释放Statement实例占用的数据库和JDBC资源，即关闭Statement实例

e. 语句PreparedStatement

- PreparedStatement接口用来执行动态的SQL语句，即包含参数的SQL语句
- executeQuery(String sql)方法：执行指定的静态SELECT语句，并返回一个永远不能为null的ResultSet实例
- executeUpdate(String sql)方法：执行指定的静态INSERT、UPDATE或DELETE语句，并返回一个int型数值，为同步更新记录的条数
- close()方法：立即释放Statement实例占用的数据库和JDBC资源，即关闭Statement实例

f. 访问结果集ResultSet

- ResultSet实例通过执行查询数据库的语句生成，类似于一个数据表，存放检索后获取的结果集。
- ResultSet实例具有指向其当前数据行的指针。最初，指针指向第一行记录的前方，通过next()方法可以将指针移动到下一行。
- ResultSet提供了从当前行检索不同类型列值的get×××()方法，均有两个重载方法，可以通过列的索引编号或者列的名称检索，索引编号从1开始。

ResultSet常用方法

方法名称	功能描述
first()	移动指针到第一行；如果结果集为空则返回false，否则返回true；如果结果集类型为TYPE_FORWARD_ONLY将抛出异常
last()	移动指针到最后一行；如果结果集为空则返回false，否则返回true；如果结果集类型为TYPE_FORWARD_ONLY将抛出异常
previous()	移动指针到上一行；如果存在上一行则返回true，否则返回false；如果结果集类型为TYPE_FORWARD_ONLY将抛出异常
next()	移动指针到下一行；指针最初位于第一行之前，第一次调用该方法将移动到第一行；如果存在下一行则返回true，否则返回false
close()	立即释放ResultSet实例占用的数据库和JDBC资源，当关闭所属的Statement实例时也执行此操作。

ResultSet常用方法

方法名称	功能描述
String getString(int)	返回ResultSet当前行数据中对应某一列号的String数据值
String getString(String)	返回ResultSet当前行数据中对应某一列名的String数据值
int getInt(int)	返回ResultSet当前行数据中对应某一列号的int数据值
int getInt(String)	返回ResultSet当前行数据中对应某一列名的int数据值
Float getFloat(int)	返回ResultSet当前行数据中对应某一列号的Float数据值
Float getFloat(String)	返回ResultSet当前行数据中对应某一列名的Float数据值

5. 数据库操作

- 查询操作
- 添加操作
- 修改操作
- 删除操作

a. 查询操作

① 加载数据库驱动
② 使用DriverManager创建一个Connection对象
③ 使用Connection对象创建一个Statement或者PreparedStatement对象
④ 使用Statement或者PreparedStatement对象执行查询（executeQuery）得到ResultSet对象
⑤ 遍历ResultSet对象
⑥ 关闭ResultSet对象
⑦ 关闭Statement或者PreparedStatement对象
⑧ 关闭连接

b. 添加、修改、删除操作

① 加载数据库驱动
② 使用DriverManager创建一个Connection对象
③ 使用Connection对象创建一个Statement或者PreparedStatement对象
④ 使用Statement或者PreparedStatement对象执行更新（executeUpdate）得到int型执行结果
⑤ 关闭Statement或者PreparedStatement对象
⑥ 关闭连接

6. MVC设计模式

- 模型-视图-控制器（Model-View-Controller，MVC）
- 即把Web应用的输入、输出和处理流程按照Model、View和Controller分成三层。
- 视图（View）用于与用户交互，可以用HTML、JSP实现。
- 模型（Model）用于表示业务数据和实现业务逻辑，可以用JavaBean类实现。
- 控制器（Controller）完成流程控制，它接收来自视图层用户输入的数据并调用相应的模型进行处理，最后选择合适的视图去响应用户，控制层可以用Servlet实现。

MVC设计模式的特点

- MVC模式完全实现了页面表示和业务逻辑的分离。
- MVC模式的优势之一就在于三层各施其职，互不干涉。
- MVC模式也有它的缺点，其一就是由于它没有明确的定义，所以完全理解MVC并不容易。
- MVC设计模式更适合大型项目的开发和管理。
- MVC成熟的框架——Struts, Spring MVC等

7. 当前Java Web开发的主流模式

- 客户端第一次访问时，服务器只向客户端提供HTML文件，形成客户端网页的主要结构
- 客户端再通过AJAX技术访问Java Servlet获取数据，从而形成最终的网页
- 优点：
  - 显示和数据相分离，便于服务器程序开发的分工
  - 便于服务器端数据提供技术的改变
  - HTML服务和数据服务相分离，减轻服务器的负担
  - 便于使用Servlet将数据提供给窗口应用程序，移动APP等其他C/S结构客户端

主流模式示意图