1. 数据库编程
2. 概述

Java数据库连接（JDBC）API在1996年发布，开发者通过这个API接口连接到数据库，并使用结构化查询语言（SQL）完成对数据库的操作。

1. JDBC驱动
2. 概述

由于多个数据库供应商提供的协议无法统一，因此Java为SQL访问提供了一套纯Java API，即与数据库无关的API，并且提供了一个驱动管理器，以允许第三方驱动程序注册到驱动管理器中，并且可以连接到特定的数据库。通过API编写的程序都可以与驱动管理器进行通信，而驱动管理器可以通过驱动程序与实际的数据库进行通信。JDBC的驱动遵循了微软的ODBC，ODBC为C语言访问数据库提供了一套编程接口。

1. 驱动分类
2. 将JDBC翻译为ODBC，使用ODBC驱动与数据库通信，即JDBC/ODBC桥。
3. 由部分Java程序和部分本地代码构成，用于与数据库的客户端API进行通信，使用这种驱动之前，不仅需要安装Java类库，还要安装平台相关的代码。
4. 纯Java客户端类库，使用一种与具体数据库无关的协议将数据库请求发送给服务器构件，然后构件将数据库请求翻译成数据库相关的协议。
5. 纯Java类库，将JDBC请求直接翻译为数据库相关的协议。
6. JDBC的典型用法

传统的C/S架构中，在服务器端部署数据库，将JDBC驱动部署在客户端。如今的三层模型中，客户端不直接调用数据库，而是调用服务器上的中间件层，由中间件层完成数据库查询操作。这种模型可以将可视化表示（客户端）从业务逻辑（中间层）和原始数据（数据库）中分类出来，即客户端可以有多种表现形式。

1. JDBC配置
2. 注册驱动器类

许多JDBC的JAR文件会自动注册驱动器类，当然我们也可以手动注册，首先找出该驱动类的全限定名，通过使用DriverManager，用两种方式来注册驱动器。一种是在Java程序中加载驱动器类，即执行注册驱动器的静态初始化器，如:

Class.forName(“org.postgresql.Driver”);

另一种方式是设置jdbc.drivers属性，可以使用命令行参数来指定这个属性，如：

java -Djdbc.drivers=org.postgresql.Driver Name

或者在应用中设置系统属性，如：

System.setProperty(“jdbc.drivers”, “org.postgresql.Driver”);

1. 使用JDBC语句
2. 创建Connection对象

首先通过DriverManager创建连接对象Connection，如下所示：

Connection conn = DriverManager.getConnection(url ,username, password);

1. 创建Statement对象

然后创建Statement语句对象，执行各种SQL语句。每个Connection对象可以创建多个Statement对象，同一个Statement对象可以用于多个不相关的命令和查询，但是一个Statement对象最多只能有一个打开的结果集。我们需要确保在一个Statement对象触发新的查询或更新语句之前结束对所有结果集的处理，因为前序查询的所有结果集都会被自动关闭。使用try(){}语句块来自动关闭对象。

Statement stat = conn.createStatement();

1. 执行SQL语句

通过Statement对象执行SQL语句。

1. executeQuery(command)

SELECT查询操作，返回一个ResultSet类型对象，可以通过它来每次一行地迭代遍历所有查询结果。

ResultSet rs = stat.executeQuery(“SELECT \* FROM Books”);

结果集中行的顺序是任意排列的，除非使用ORDER BY 子句指定行的顺序，否则行顺序是没有意义的。获取结果集后，通过访问器方法来获取其中的信息，每个访问器都有两种形式，一种接受数字型参数，另一种接受字符串参数，数据库的列序号是从1开始计算的。

String isbn = rs.getString(1);

double price = rs.getDouble(“Price”);

1. 执行查询操作
2. 预备语句

我们没有必要在每次进行查询时都建立新的查询语句，可以准备一个带有宿主变量的查询语句，每次查询时只需要为该变量填入不同的字符串就可以反复使用。每个宿主变量使用“？”来表示，如下所示：

String statement = “SELECT \* FROM tb\_name WHERE id=?”;

PreparedStatement stat = conn.prepareStatement(statement);

执行预备语句之前，必须使用set方法将变量绑定到实际的值上，第一个参数是宿主变量的位置，即第几个变量，第二个参数为变量的值。

stat.setString(1, “3”);

由于字符串拼接有潜在的危险，如引号这样的字符或者用户的输入可能会导致的注入攻击，因此只要查询涉及变量就应该使用预备语句。当Connection对象关闭后，相关的预备语句也就失效了。

1. 读写LOB

数据还可以存储大对象，如图片或者其他数据，在SQL中，二进制大对象称为BLOB，字符型大对象称为CLOB。要读取LOB，需要执行SELECT语句，然后在ResultSet上调用getBlob或getClob方法，这样就可以获得Blob或者Clob类型的对象了。

try(ResultSet result = stat.executeQuery()){

if(result.next()){

Blob coverBlob = result.getBlob();

Image coverImage = ImageIO.read(coverBlob.getBinaryStream());

}

}

要将LOB置于数据库中，需要在Connection对象上调用createBlob或者createBlob，然后获取一个用于该LOB的输出流或写出器，写出数据，并将该对象存储到数据库中。

1. 获取自动生成的键

大多数数据库都支持在数据库中对行自动编号的机制，这些自动编号的值通过作为主键，尽管JDBC没有提供独立于提供商的自动生成键的解决方案，但是提供了获取自动生成建的途径，当插入一个新行，该行的键自动生成时，我们可以这样来获取：

stat.executeUpdate(insertStatement, Statement.RETURN\_GENERATED\_KEYS);

ResultSet rs = stat.getGeneratedKeys();

if(rs.next()){

int key = rs.getInt(1);

}

1. 可滚动和可更新的结果集
2. 可滚动的结果集

默认的结果集是不可滚动和不可更新的，可以通过Statement对象的设置来获取不同的结果集。

Statement stat = conn.createStatement(type, concurrency);

type有三种类型：

1. TYPE\_FORWARD\_ONLY

不能滚动

1. TYPE\_SCROLL\_INSENSITIVE

可以滚动，但对数据库变化不敏感，即不会实时更新数据库。

1. TYPE\_SCROLL\_SENSITIVE

可以滚动，对数据库变化敏感。

concurrency两种类型：

1. CONCUR\_READ\_ONLY

结果集不能用于更新数据库。

1. CONCUR\_UPDATABLE

结果集可以用于更新数据库。

1. 可更新的结果集

如果希望编辑结果集中的数据，并且将结果集中的数据变更自动反应到数据库中，那么需要使用可更新的结果集。

Statement stat = conn.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

1. 行集

客户端与数据库的连接并不一定是高效的，很多时候连接资源是浪费的，因此我们使用行集来解决这个问题。RowSet接口扩展自ResultSet接口，但无需使用保持与数据库的链接。

1. 构建行集

RowSetFactory fac = RowSetProvider.newFactory();

CachedRowSet crs = fac.createCachedRowSet();

有好几种行集类型，都扩展了RowSet接口。

1. 被缓存的行集

一个被缓存的行集中包含了一个结果集中所有的数据，断开数据库后依然可以使用行集，同时可以修改行集中的数据，再通过显示的请求发送到数据库中更新修改。

1. 元数据

JDBC还提供了关于数据库以及其表结构的详细信息，即元数据，区别于数据库中存储的实际数据，我们可以获取三类元数据，一类是数据库的元数据，一类是结果集的元数据，一类是预备语句参数的元数据。通过从数据库连接中获取一个DatabaseMetaData对象来获取元数据。

1. 事务
2. 概述

一组语句构成一个事务，当所有语句都顺利执行后，事务可以被提交，否则如果其中某个语句遇到错误，那么事务将被回滚，就好像没有语句被执行过一样。

1. JDBC事务编程

默认情况下，数据库连接处于自动提交模式，即每个SQL语句一旦被执行就会被提交给数据库，一旦命令被提交，就无法对它进行回滚操作，在使用事务时，要关闭这个默认值。

conn.setAutoCommit(false);

conn.commit();

conn.rollback();

1. 保存点

在使用某些驱动程序时，使用保存点来更细粒度地控制回滚操作，创建一个保存点即意味着只需要返回到这个点，而不是放弃整个事务。当不需要保存点时，释放它。

Statement stat = conn.createStatement();

state.executeUpdate(command1);

Savepoint svpt = conn.setSavepoint();

stat.execcuteUpdate(command2);

if(…) conn.rollback(svpt);

conn.commit();

conn.releaseSavepoint(svpt);

1. 数据库连接池机制

JDBC规范中为实现者提供了用以实现连接池服务的手段，但是JDK并没有实现数据库连接池，JDBC驱动中通常也不包含这项服务，相反，WEB容器和应用服务器的开发商会提供连接池服务的实现。连接池的机制通常还可以用于预备语句上。