Java 数据库连接

约瑟夫·哈利特

2023 年 2 月 13 日

这是关于什么的?

以前,当使用 SQL 时,我们将它当作自己的东西来使用……

▶直接在 SQL 中运行查询▶将响应显示为表格

在现实世界中,我们很少想单独访问数据库

▶相反,它作为程序的一部分在编程语言中使用

不同的语言对于不同的数据库有不同的API……

▶…但是 Java 拥有几乎所有这些的JDBC

数据库连接

▶库位于 java.sql 和 javax.sql 包中▶将所有数据库功能
包装到看起来很像Oracle SQL 的东西中。 ▶支持准备好的语句（您想要使用这些）

它是什么样子的？

```
导入 java.sql.*;

尝试 (最终连接 conn = DriverManager.getConnection(    jdbc:sqlite:database.db    )) { conn.createStatement()

                        .executeQuery(    创建表用户(用户名 TEXT PRIMARY KEY,密码 TEXT)    );
} catch (最终的 SQLException err)
      { System.out.println(err);
}
```

java.sql.SQLException：找不到适用于 jdbc:sqlite:database.db 的驱动程序

当您找到合适的驱动程序并将其添加到您的 CLASSPATH 中时……

```
导入 java.sql.*;

try (final Connection conn = DriverManager.getConnection(    jdbc:sqlite:database.db    ))

        { conn.createStatement() .executeQuery(    CREATE TABLE users(用户名 TEXT PRIMARY KEY, 密码
TEXT)    ); } catch (最终的 SQLException
        err) { System.out.println(err);
}
```

org.sqlite.SQLiteException: [SQLITE_ERROR] SQL 错误或丢失数据库（表用户已存在）

让我们添加一些合适的用户...

```
导入 java.sql.*;导入
java.util.*;

最终 var users = new HashMap<String, String>();
users.put(  约瑟夫  ,  密码  );
users.put(  马特  ,  密码1  );
users.put(  帕萨  ,  12345  );

尝试 (最终连接 conn = DriverManager.getConnection(  jdbc:sqlite:database.db  )) {
    conn.createStatement().executeUpdate(  从用户中删除  );最终 var 语
    句 = conn.prepareStatement(  INSERT INTO users VALUES(?, ?)  ); for (final var user :
    users.keySet()) { statements.setString(1,
        user); } statements.setString(2,
        users.get(user));语句.executeUpdate();

    }
} catch (最终的 SQLException err)
    { System.out.println(err);
}
```

并将它们列出来……

```
导入 java.sql.*;导入
java.util.*;

System.out.println(    |用户|密码    );尝试 (最终连接 conn
= DriverManager.getConnection(    jdbc:sqlite:database.db    )) {
        最终 var results =
                conn.createStatement() .executeQuery(    SELECT
        * FROM users    ); while (结果.next())
                System.out.println(    |    +results.getString(1) +    |
                                        +results.getString(2)); } catch (最
终的 SQLException err)
        { System.out.println(err);
}
```

用户 密码 密码1

马特

约瑟夫密码

帕塔12345

为什么不是这个……

添加所有用户时,我们使用PreparedStatement 来添加所有用户。最终 var 语句 =

```
conn.prepareStatement(    INSERT INTO users VALUES(?, ?)    ); for (final var user : users.keySet())
{ statements.setString(1, user); }
        statements.setString(2,
        users.get(user));语句.executeUpdate();

}
```

这不是更容易吗?

```
for (final var user : users.keySet())

        conn.createStatement() .executeUpdate(    INSERT INTO users    +    VALUES (       +user+       ,       +users.get(user
```

SQL注入

这会导致一个可怕的漏洞,称为注入攻击▶您也可以使用 shellscript 执行类似的操作 ;-) ▶如果您有兴趣,请搜索Shellshock 漏洞...

准备好的语句的作用是确保您添加的内容与您所说的一致假设您对登录代码执行类似的操作:

从用户中选择用户名
WHERE 用户名 = “约瑟夫”
和密码=“密码”;

用户名
约瑟夫

假设用户名和密码取自网站登录表单... ▶如果我尝试使用以下密码登录会发生什么:

OR 1 OR 密码 = 呵呵

坏事

附上一份准备好的声明：

从用户中选择用户名
WHERE 用户名 = “约瑟夫”
和密码 = OR 1 OR 密码 = 呵呵 ；

没有准备好的声明：

从用户中选择用户名
WHERE 用户名 = “约瑟夫”
和密码 = OR 1 OR 密码 = 呵呵 ；

用户名
马特
约瑟夫
帕塔

始终使用准备好的语句现在编译器甚至会发出有关此的警告和错误.....

▶或者findbugs将...

交易

JDBC 使事务变得简单的另一件很酷的事情……
假设您想要对数据库进行大量添加和更新……

- 如果中间出了问题怎么办？

您可以手动回滚您添加的所有新数据和所做的更改……

- 听起来很乏味
- 让我们自动化吧！

交易流程

1.开始新的交易

2.做好你的工作3.完成后全力以赴

4.发生错误时回滚

请问用Java吗？

```
导入 java.sql.*;导入
java.util.*;

尝试 (最终连接 conn = DriverManager.getConnection(    jdbc:sqlite:database.db    )) {
    conn.setAutoCommit(假);最终
    var save = conn.setSavepoint();尝试

    { conn.createStatement() .executeQuery(    INSERT INTO users VALUES (    Alice    ,    pa55w0rd    )    );
        conn.createStatement() .executeQuery(    INSERT INTO users VALUES (    Bob    ,    Pa55w0Rd7    )    );
        if (true) 抛出新的异常(“哎呀！”);
        conn.createStatement() .executeQuery(    INSERT INTO users VALUES (    Eve    ,    backd00r    )    );
        conn.commit(); }
    catch (最终异常错误)
        { conn.rollback(save); }
    最后
        { conn.setAutoCommit(true);
    }
} catch (最终的 SQLException err)
    { System.out.println(err);
}
```

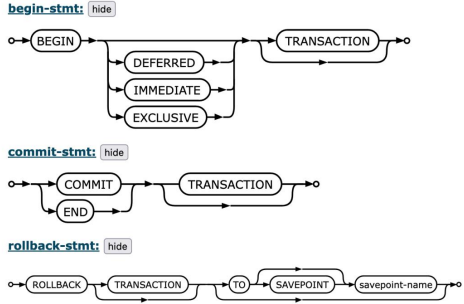
现在如果我们查询用户……

从用户中选择*；

用户名 密码 密码1
马特密
约瑟夫12345
帕塔

我们的表保持不变……整个事务被回滚。

（哦，顺便说一句，SQLite 也可以在 SQL 中执行事务）



返回实体关系图...

回到数据库运行视频的开始,我们只是在涂鸦图表而不是构建数据库▶ Java 类看起来非常像一个实体

如果我们可以上一堂课并让数据库导入和保存全部为我们处理,这不是很好吗?

冬眠！

https://hibernate.org在
JDBC之上构建来做到这一点！ ▶注释您的
类▶编写一堆 XML 来告诉它您
的数据库格式▶魔法和稍微高级的查询语言

我们将在实验室里玩它……

结论

JDBC 允许您从 Java 访问 SQL

▶确保加载正确的驱动程序▶捕获 SQLExceptions

▶使用准备好的语句和事务来防

止错误▶如果您愿意,还可以使用像Hibernate这样的 ORM 。

重要提示请不要像我们在讲座中那样

实际实现密码存储.....

▶首先去与网络或加密货币团体中的某人交谈... ▶或先阅读 NIST 800-63

如果你愿意,我会写关于你的论文;-)

约瑟夫·哈利特、尼基尔·帕特奈克、本杰明·什里夫和阿瓦斯·拉希德。 “做这个!这样做!,什么都不会发生”规范是否会导致安全存储密码? 2021 年 IEEE/ACM 第 43 届国际软件工程会议 (ICSE)。 2021 年。