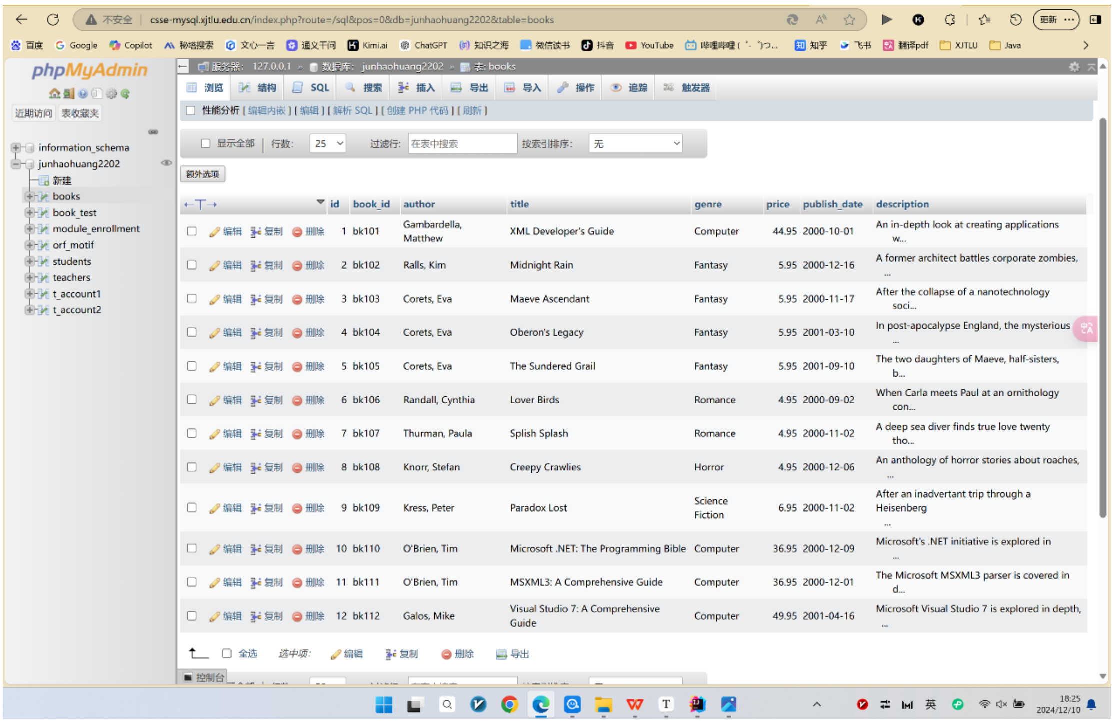
The result of the database



id	book_id	author	title	genre	price	publish_date	description
1	bk101	Gambardella, Matthew	XML Developer's Guide	Computer	44.95	2000-10-01	An in-depth look at creating applications w...
2	bk102	Ralls, Kim	Midnight Rain	Fantasy	5.95	2000-12-16	A former architect battles corporate zombies, ...
3	bk103	Corets, Eva	Maeve Ascendant	Fantasy	5.95	2000-11-17	After the collapse of a nanotechnology soci...
4	bk104	Corets, Eva	Oberon's Legacy	Fantasy	5.95	2001-03-10	In post-apocalypse England, the mysterious ...
5	bk105	Corets, Eva	The Sundered Grail	Fantasy	5.95	2001-09-10	The two daughters of Maeve, half-sisters, b...
6	bk106	Randall, Cynthia	Lover Birds	Romance	4.95	2000-09-02	When Carla meets Paul at an ornithology con...
7	bk107	Thurman, Paula	Splish Splash	Romance	4.95	2000-11-02	A deep sea diver finds true love twenty tho...
8	bk108	Knorr, Stefan	Creepy Crawlies	Horror	4.95	2000-12-06	An anthology of horror stories about roaches, ...
9	bk109	Kress, Peter	Paradox Lost	Science Fiction	6.95	2000-11-02	After an inadvertant trip through a Heisenberg ...
10	bk110	O'Brien, Tim	Microsoft .NET: The Programming Bible	Computer	36.95	2000-12-09	Microsoft's .NET initiative is explored in ...
11	bk111	O'Brien, Tim	MSXML3: A Comprehensive Guide	Computer	36.95	2000-12-01	The Microsoft MSXML3 parser is covered in d...
12	bk112	Galos, Mike	Visual Studio 7: A Comprehensive Guide	Computer	49.95	2001-04-16	Microsoft Visual Studio 7 is explored in depth, ...

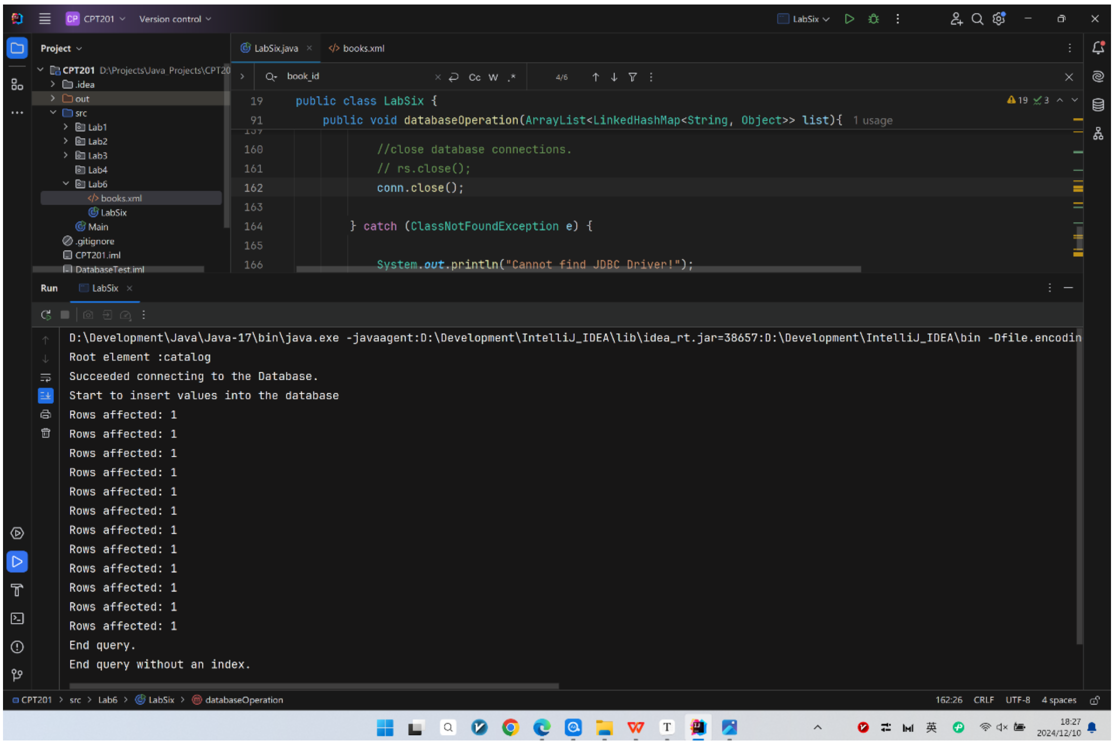
phpMyAdmin
服务器: 127.0.0.1 » 数据库: junhaohuang2202 » 表: books
浏览
结构
SQL
搜索
插入
导出
导入
操作
追踪
触发器
性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]
显示全部 | 行数: 25 过滤行: 在表中搜索 按索引排序: 无
额外选项
编辑 复制 删除
全选 选中项: 编辑 复制 删除 导出
控制台
information_schema
junhaohuang2202
新建
books
book_test
module_enrollment
orf_motif
students
teachers
t_account1
t_account2
18:25
2024/12/10

The code can successfully get the result above:

```
package Lab6;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.File;
import java.sql.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.LinkedHashMap;

public class LabSix {

    // initiate database information
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    //replace the following three parameter values with your own ones
    static final String DB_URL = "jdbc:mysql://10.7.1.127/junhaohuang2202";
    static final String USER = "JunhaoHuang2202";
    static final String PASS = "123";

    public ArrayList<LinkedHashMap<String, Object>> extractInfo(String filepath)
{
        ArrayList<LinkedHashMap<String, Object>> list = new ArrayList<>();
        try {
            File inputFile = new File("src/Lab6/books.xml");

            DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
            Document doc = dBuilder.parse(inputFile);
            doc.getDocumentElement().normalize();
            System.out.println("Root element :" +
doc.getDocumentElement().getNodeName());
            NodeList nList = doc.getElementsByTagName("book");

            // get each book info
            for (int temp = 0; temp < nList.getLength(); temp++) {
                LinkedHashMap<String, Object> entity = new LinkedHashMap<>();
                Node nNode = nList.item(temp);
                // print book
                // System.out.println("\nCurrent Element:" + nNode.getNodeName()
+ temp);

                if (nNode.getNodeType() == Node.ELEMENT_NODE) {
                    Element eElement = (Element) nNode;
                    // get book_id
                    String book_id = eElement.getAttribute("id");

                    // get author
                    String author =
eElement.getElementsByTagName("author").item(0).getTextContent();

                    // get title
                    String title =
eElement.getElementsByTagName("title").item(0).getTextContent();

                    // get genre
                    String genre =
eElement.getElementsByTagName("genre").item(0).getTextContent();

                    // get price
                    String                        price                        =
eElement.getElementsByTagName("price").item(0).getTextContent();

                    // get date
                    String                        date                         =
eElement.getElementsByTagName("publish_date").item(0).getTextContent();

                    // get description
                    String                     description                     =
eElement.getElementsByTagName("description").item(0).getTextContent();

                    // add to map
                    entity.put("book_id", book_id);
                    entity.put("author", author);
                    entity.put("title", title);
                    entity.put("genre", genre);
                    entity.put("price", price);
                    entity.put("date", date);
                    entity.put("description", description);

                    // 讲当前的entity添加到arraylist当中
                    list.add(entity);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return list;
    }

    public   void   databaseOperation(ArrayList<LinkedHashMap<String,   Object>>
list){
        Connection conn;
        Statement stmt;
        ResultSet rs;
        String sql;
        String first;
        long begin;
        long end;

        try {
            Class.forName(JDBC_DRIVER);

            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            if (!conn.isClosed()) {
                System.out.println("Succeeded connecting to the Database.");
            }

            stmt = conn.createStatement();

            //prepare query without an index
            System.out.println("Start to insert values into the database");
            for (int i = 0; i < list.size(); i++) {
                LinkedHashMap<String, Object> entity = list.get(i);
                try {
                    String priceStr = (String) entity.get("price");
                    String dateStr = (String) entity.get("date");
                    String bookId = (String) entity.get("book_id");
                    String author = (String) entity.get("author");
                    String title = (String) entity.get("title");
                    String genre = (String) entity.get("genre");
                    String description = (String) entity.get("description");

                    // check null value
                    if (priceStr == null || dateStr == null || bookId == null ||
author == null || title == null || genre == null || description == null) {
                        System.out.println("Skipping incomplete data: " +
entity);
                        continue;
                    }

                    // convert float and date
                    float price = Float.parseFloat(priceStr);
                    DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("yyyy-MM-dd");
                    LocalDate localDate = LocalDate.parse(dateStr, formatter);
                    java.sql.Date sqlDate = java.sql.Date.valueOf(localDate);

                    // 构建 SQL
                    sql = "INSERT INTO books (book_id, author, title, genre,
price, publish_date, description) VALUES (?, ?, ?, ?, ?, ?, ?)";
                    try (PreparedStatement pstmt = conn.prepareStatement(sql))
{
                        pstmt.setString(1, bookId);
                        pstmt.setString(2, author);
                        pstmt.setString(3, title);
                        pstmt.setString(4, genre);
                        pstmt.setFloat(5, price);
                        pstmt.setDate(6, sqlDate);
                        pstmt.setString(7, description);

                        int affectedRows = pstmt.executeUpdate();
                        System.out.println("Rows affected: " + affectedRows);
                    }
                } catch (SQLException e) {
                    System.err.println("SQL error: " + e.getMessage());
                    e.printStackTrace();
                } catch (Exception e) {
                    System.err.println("Error: " + e.getMessage());
                    e.printStackTrace();
                }
            }
            System.out.println("End query.");
            conn.close();

            //close database connections.
            conn.close();

        } catch (ClassNotFoundException e) {

            System.out.println("Cannot find JDBC Driver!");
            e.printStackTrace();

        } catch (SQLException e) {

            e.printStackTrace();

        } catch (Exception e) {

            e.printStackTrace();

        }
    }

    public static void main(String[] args) {
        LabSix labSix = new LabSix();
        ArrayList<LinkedHashMap<String,       Object>>       result        =
labSix.extractInfo("books.xml");

        for (LinkedHashMap<String, Object> entity : result) {
            System.out.printf((String) entity.get("book_id"));
        }
```

```
            labSix.databaseOperation(result);
        }



}
```

The screen shot of terminal output



```
public class LabSix {
    public void databaseOperation(ArrayList<LinkedHashMap<String, Object>> list){
            //close database connections.
            // rs.close();
            conn.close();

        } catch (ClassNotFoundException e) {

            System.out.println("Cannot find JDBC Driver!");
```

```
D:\Development\Java\Java-17\bin\java.exe -javaagent:D:\Development\IntelliJ_IDEA\lib\idea_rt.jar=38657:D:\Development\IntelliJ_IDEA\bin -Dfile.encodin
Root element :catalog
Succeeded connecting to the Database.
Start to insert values into the database
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
Rows affected: 1
End query.
End query without an index.
```