

# XML

GAM/IT/2022/F/0103

Ashintha Gaurawa

## XML PARSES

```
package books;

import org.w3c.dom.*;
import javax.xml.parsers.*;

public class XmlParser {
    public static void main(String[] args) {
        try {
            // Create a new DocumentBuilderFactory and DocumentBuilder
            DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();

            // Parse the XML file
            Document document = builder.parse("C:\\Users\\DELL\\Desktop\\BookXML-
            main\\src\\books\\books.xml");
            //Document document = builder.parse("C:\\Users\\User\\Desktop\\books.xml\\books.xml");

            // Normalize the document
            document.getDocumentElement().normalize();

            // Get the root element (library)
            NodeList nodeList = document.getElementsByTagName("book");

            // Loop through each book in the XML document
            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);

                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;

                    // Get and print the details of each book
                    String title =element.getElementsByTagName("title").item(0).getTextContent();
                    String author =element.getElementsByTagName("author").item(0).getTextContent();
                    String year =element.getElementsByTagName("year").item(0).getTextContent();
                    String genre =element.getElementsByTagName("genre").item(0).getTextContent();
```

```

System.out.println("Title: " + title);
System.out.println("Author: " + author);
System.out.println("Year: " + year);
System.out.println("Genre: " + genre);
System.out.println("-----");
}
}

} catch (Exception e) {
e.printStackTrace();}}}

```

## Book.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
Click nbfs://nbhost/SystemFileSystem/Templates/XML/XMLDocument.xml to edit this template
-->

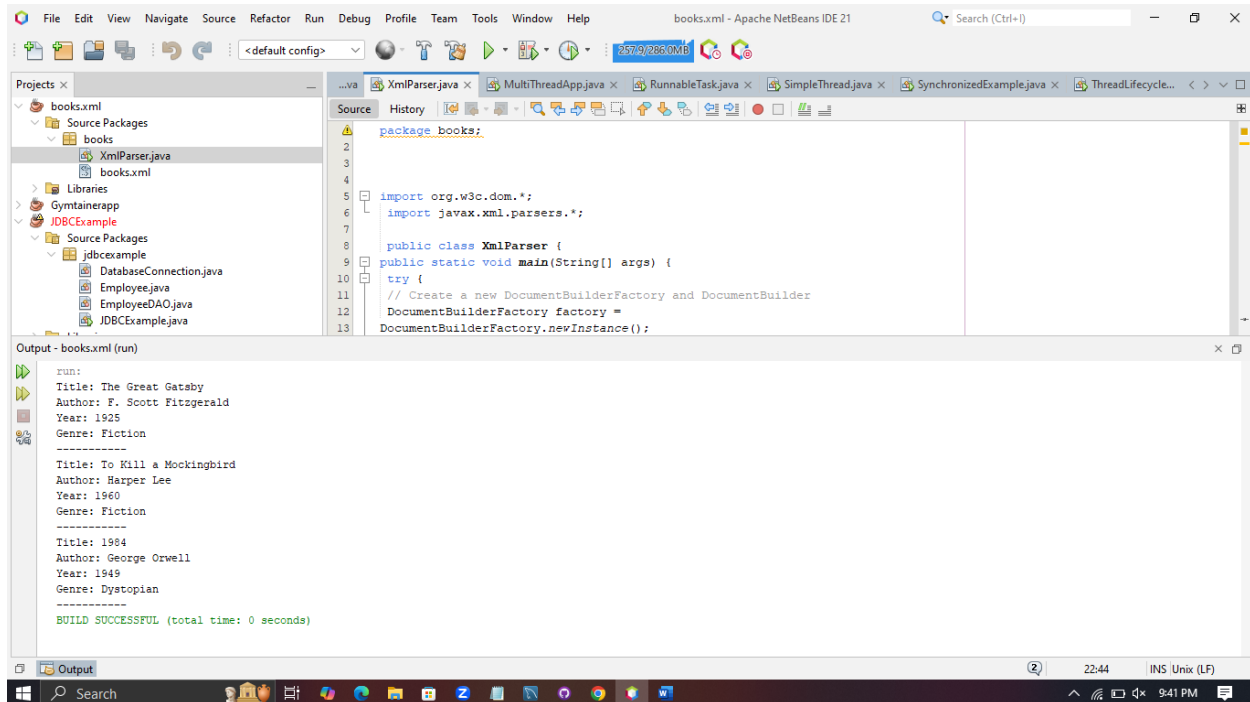
```

```

<library>
  <book>
    <title>The Great Gatsby</title>
    <author>F. Scott Fitzgerald</author>
    <year>1925</year>
    <genre>Fiction</genre>
  </book>
  <book>
    <title>To Kill a Mockingbird</title>
    <author>Harper Lee</author>
    <year>1960</year>
    <genre>Fiction</genre>
  </book>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <year>1949</year>
    <genre>Dystopian</genre>
  </book>
</library>

```

# OUTPUT



# JDBC

## DatabaseConnection

```
package jdbcexample;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/employee_db"; // Database URL
    private static final String USER = "root"; // Your MySQL username
    private static final String PASSWORD = "ashi2002"; // Your MySQL password

    public static Connection getConnection() throws SQLException {
        try {
            // Load the JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");
            // Return the database connection
            return DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("Connection failed: " + e.getMessage());
            throw new SQLException("Failed to establish connection.");
        }
    }
}
```

## Employee

```
package jdbcexample;

public class Employee {
    private int id;
    private String name;
    private String position;
    private double salary;

    public Employee(int id, String name, String position, double salary) {
        this.id = id;
        this.name = name;
    }
}
```

```

        this.position = position;
        this.salary = salary;
    }

    // Getters and setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getPosition() { return position; }
    public void setPosition(String position) { this.position = position; }

    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }

    @Override
    public String toString() {
        return "Employee{id=" + id + ", name=" + name + ", position=" +
            position + ", salary=" + salary + '}';
    }
}

```

## EmployeeDAO

```

package jdbcexample;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class EmployeeDAO {

    // Create an employee
    public static void addEmployee(String name, String position, double salary)
    {
        String sql = "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";

        try (Connection conn = DatabaseConnection.getConnection(); PreparedStatement stmt =
            conn.prepareStatement(sql)) {

            stmt.setString(1, name);
            stmt.setString(2, position);
            stmt.setDouble(3, salary);

```

```

        int rowsAffected = stmt.executeUpdate();
        System.out.println("Employee added successfully. Rows affected: " + rowsAffected);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Read all employees
public static List<Employee> getAllEmployees() {
    List<Employee> employees = new ArrayList<>();
    String sql = "SELECT * FROM employees";

    try (Connection conn = DatabaseConnection.getConnection(); Statement stmt =
conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {

        while (rs.next()) {
            Employee employee = new Employee(

                rs.getInt("id"),
                rs.getString("name"),
                rs.getString("position"),
                rs.getDouble("salary")
            );
            employees.add(employee);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return employees;
}

// Update an employee's information
public static void updateEmployee(int id, String name, String position,
double salary) {
    String sql = "UPDATE employees SET name = ?, position = ?, salary = ? WHERE id = ?";

    try (Connection conn = DatabaseConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setString(1, name);
        stmt.setString(2, position);
        stmt.setDouble(3, salary);
        stmt.setInt(4, id);

        int rowsAffected = stmt.executeUpdate();
        System.out.println("Employee updated successfully. Rows affected: "
+ rowsAffected);
    } catch (SQLException e) {

```

```

        e.printStackTrace();
    }
}

// Delete an employee
public static void deleteEmployee(int id) {
    String sql = "DELETE FROM employees WHERE id = ?";

    try (Connection conn = DatabaseConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {

        stmt.setInt(1, id);
        int rowsAffected = stmt.executeUpdate();
        System.out.println("Employee deleted successfully. Rows affected: "
+ rowsAffected);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

## JDBCExample

```

package jdbcexample;

import java.util.List;

public class JDBCExample {

    public static void main(String[] args) {
        // Add employees
        EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
        EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);

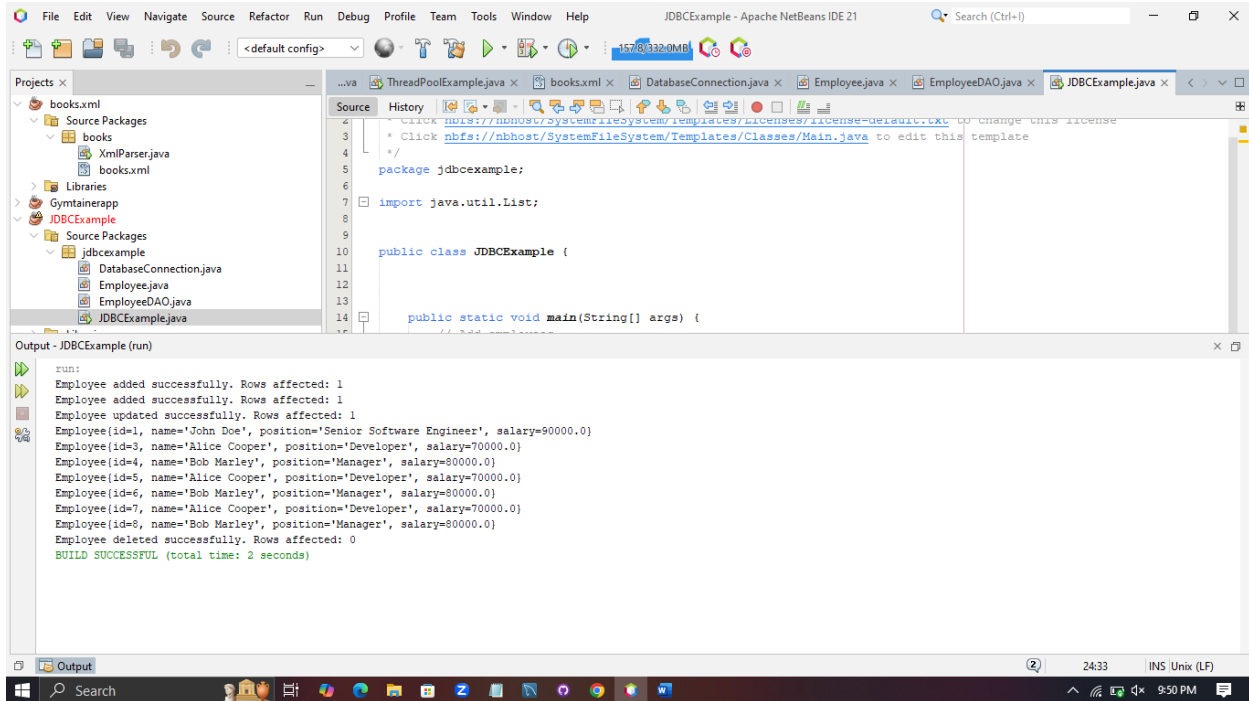
        // Update employee
        EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer",
90000);

        // Get all employees
        List<Employee> employees = EmployeeDAO.getAllEmployees();
        employees.forEach(System.out::println);

        // Delete employee
        EmployeeDAO.deleteEmployee(2);
    }
}

```

# OUTPUT



The screenshot displays the NetBeans IDE interface. The top menu bar includes File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, and Help. The toolbar shows various icons for file operations and running code. The Projects window on the left shows a project named 'JDBCExample' with a source package 'jdbcexample' containing files like DatabaseConnection.java, Employee.java, EmployeeDAO.java, and JDBCExample.java. The main editor window shows the source code of 'JDBCExample.java'. The code includes a package declaration, an import statement for 'java.util.List', and a public class 'JDBCExample' with a static void 'main' method. The output window at the bottom shows the results of running the program, including messages for adding, updating, and deleting employees, and a final 'BUILD SUCCESSFUL' message.

```
4  * Click https://www.netbeans.org/licenses/1.6.0/changes.html to change this license
5  */
6  package jdbcexample;
7
8  import java.util.List;
9
10 public class JDBCExample {
11
12
13
14     public static void main(String[] args) {
15         // Add employees
```

run:  
Employee added successfully. Rows affected: 1  
Employee added successfully. Rows affected: 1  
Employee updated successfully. Rows affected: 1  
Employee(id=1, name='John Doe', position='Senior Software Engineer', salary=90000.0)  
Employee(id=3, name='Alice Cooper', position='Developer', salary=70000.0)  
Employee(id=4, name='Bob Marley', position='Manager', salary=80000.0)  
Employee(id=5, name='Alice Cooper', position='Developer', salary=70000.0)  
Employee(id=6, name='Bob Marley', position='Manager', salary=80000.0)  
Employee(id=7, name='Alice Cooper', position='Developer', salary=70000.0)  
Employee(id=8, name='Bob Marley', position='Manager', salary=80000.0)  
Employee deleted successfully. Rows affected: 0  
BUILD SUCCESSFUL (total time: 2 seconds)



# **JAVA THREAD**

## RunnableTask

```
package multithreadapp;

/**
 *
 * @author User
 */
public class RunnableTask implements Runnable {
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getId() + " is executing the runnable task.");
    }
    public static void main(String[] args) {
        RunnableTask task1 = new RunnableTask();
        RunnableTask task2 = new RunnableTask();

        Thread thread1 = new Thread(task1);
        Thread thread2 = new Thread(task2);

        thread1.start(); // Starts thread1
        thread2.start(); // Starts thread2
    }
}
```

## SimpleThread

```
package multithreadapp;

    public class SimpleThread extends Thread {
        @Override
        public void run() {
            System.out.println(Thread.currentThread().getId() + " is executing the thread.");
        }
        public static void main(String[] args) {

            SimpleThread thread1 = new SimpleThread();
            SimpleThread thread2 = new SimpleThread();

            thread1.start(); // Starts thread1
            thread2.start(); // Starts thread2
        }
    }
```

## SynchronizedExample

```
package multithreadapp;

class Counter {
    private int count = 0;
    // Synchronized method to ensure thread-safe access to the counter
    public synchronized void increment() {
        count++;
    }
    public int getCount() {
        return count;
    }
}

public class SynchronizedExample extends Thread {
    private Counter counter;
    public SynchronizedExample(Counter counter) {
        this.counter = counter;
    }
    @Override
    public void run() {
        for (int i = 0; i < 1000; i++) {
            counter.increment();
        }
    }

    public static void main(String[] args) throws InterruptedException {
        Counter counter = new Counter();

        // Create and start multiple threads
        Thread thread1 = new SynchronizedExample(counter);
        Thread thread2 = new SynchronizedExample(counter);
        thread1.start();
        thread2.start();
        // Wait for threads to finish
        thread1.join();
        thread2.join();
        System.out.println("Final counter value: " + counter.getCount());
    }
}
```

## ThreadLifeCycleExample

```
package multithreadapp;

public class ThreadLifecycleExample extends Thread {
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName() + " - State: " +
            Thread.currentThread().getState());
    }
}
```

```

try {
    Thread.sleep(2000); // Simulate waiting state
} catch (InterruptedException e) {
    e.printStackTrace();
}
System.out.println(Thread.currentThread().getName() + " - State after sleep: " +
Thread.currentThread().getState());
}
public static void main(String[] args) {
    ThreadLifecycleExample thread = new ThreadLifecycleExample();
    System.out.println(thread.getName() + " - State before start: " +
thread.getState());
    thread.start(); // Start the thread
    System.out.println(thread.getName() + " - State after start: " +
thread.getState());
}
}

```

## ThreadPoolExample

```

package multithreadapp;

```

```

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
class Task implements Runnable {
    private int taskId;
    public Task(int taskId) {
        this.taskId = taskId;
    }
    @Override
    public void run() {
        System.out.println("Task " + taskId + " is being processed by " +
Thread.currentThread().getName());
    }
}
public class ThreadPoolExample {
    public static void main(String[] args) {
        // Create a thread pool with 3 threads
        ExecutorService executorService = Executors.newFixedThreadPool(3);
        // Submit tasks to the pool
        for (int i = 1; i <= 5; i++) {
            executorService.submit(new Task(i));
        }
        // Shutdown the thread pool
        executorService.shutdown();
    }
}

```

# OUTPUT

