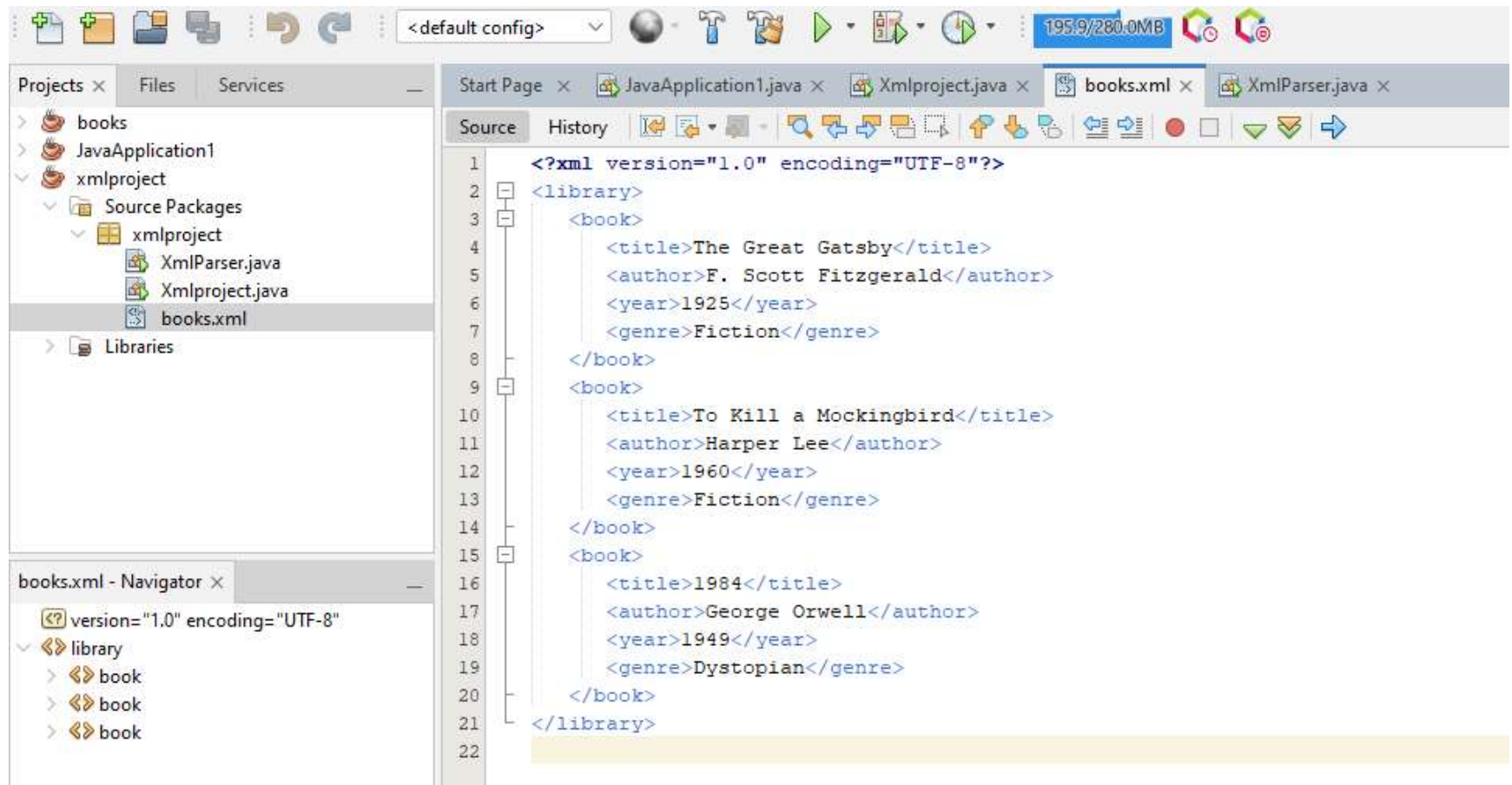


- **Creating First XML Document**

- Create a new file named books.xml



<?xml version="1.0" encoding="UTF-8"?>

<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>

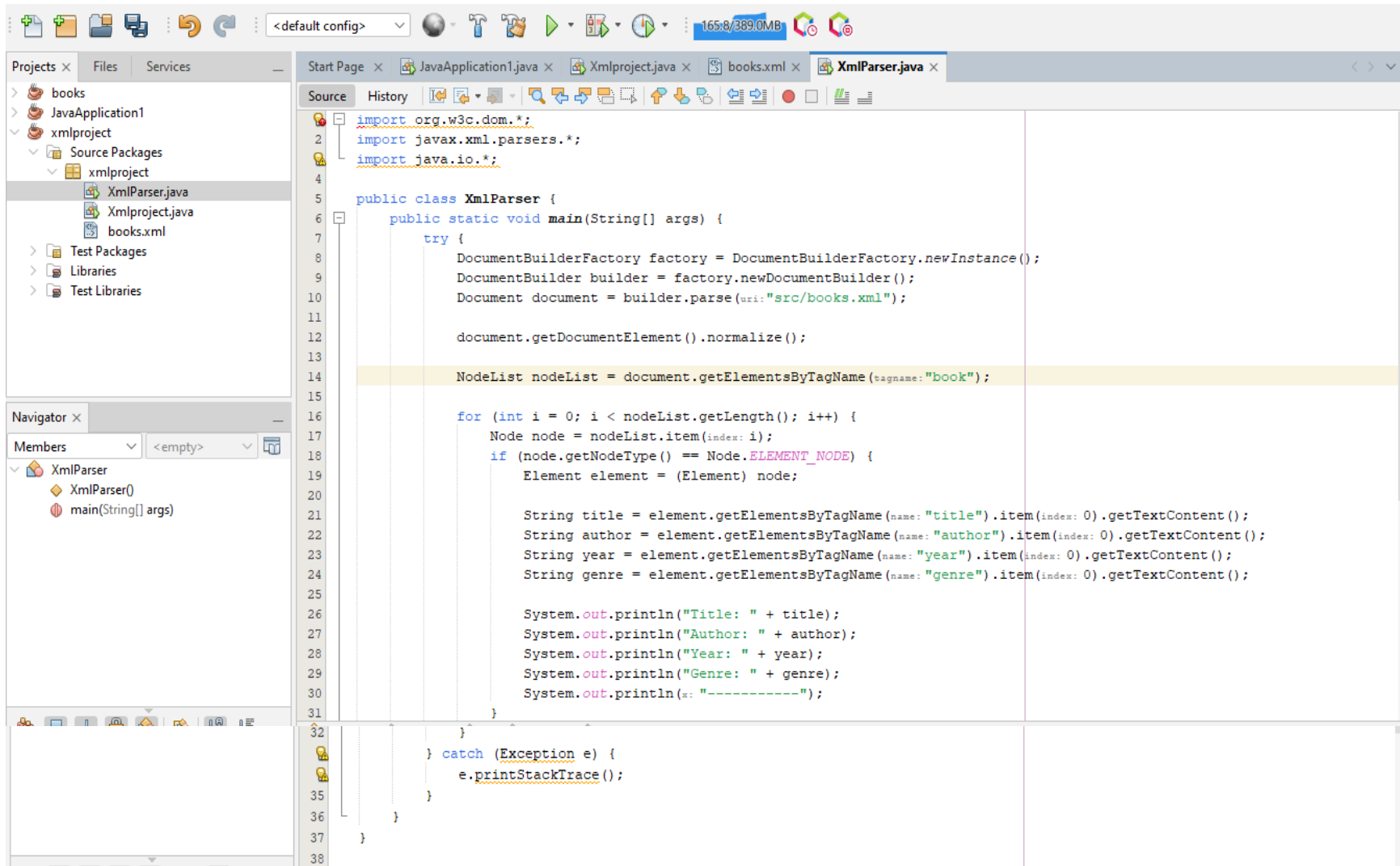
<year>1949</year>

<genre>Dystopian</genre>

</book>

</library>

- Parsing XML in Java
  - Create a Java Class for XML Parsing:



```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;

public class XmlParser {
    public static void main(String[] args) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(new File("src/books.xml"));

            document.getDocumentElement().normalize();

            NodeList nodeList = document.getElementsByTagName("book");

            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    Element element = (Element) node;

                    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();
        }
    }
}
```

```
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("books.xml");

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

            // Get all book elements
            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();  
}  
}
```

## Output

The screenshot displays an IDE interface with the following components:

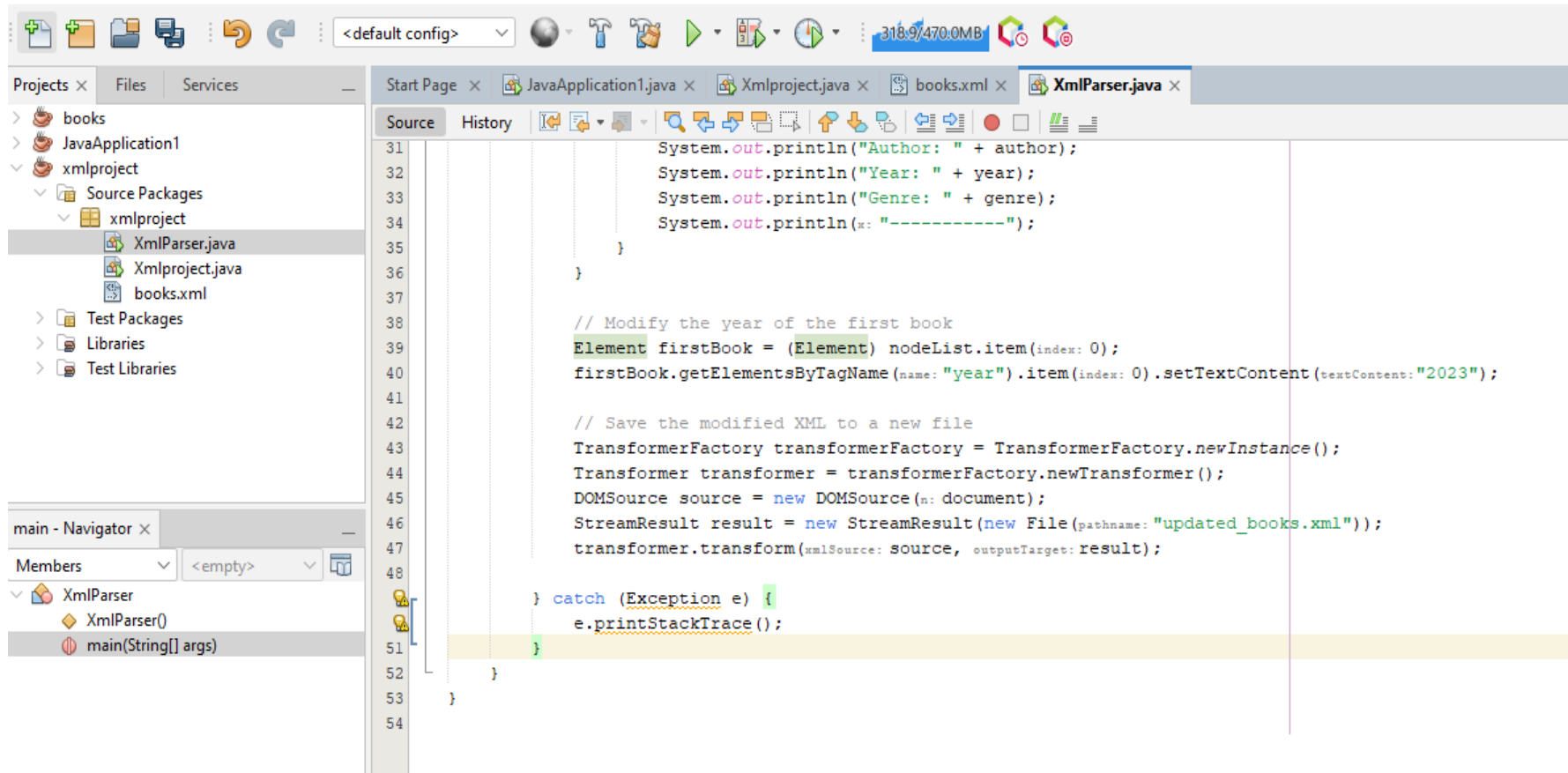
- Projects View:** Shows a project named 'xmlproject' containing 'XmlParser.java', 'Xmlproject.java', and 'books.xml'.
- Source Editor:** Displays the code in 'XmlParser.java' with line numbers 8 to 18. The code uses DocumentBuilderFactory to parse 'books.xml' and extract book details.
- main - Navigator:** Shows the 'main' method of the 'XmlParser' class.
- Output Console:** Shows the execution output, listing book details for 'The Great Gatsby', 'To Kill a Mockingbird', and '1984', followed by a successful build message.

```
8      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
9      DocumentBuilder builder = factory.newDocumentBuilder();
10     Document document = builder.parse(new URI("src/xmlproject/books.xml"));
11
12     document.getDocumentElement().normalize();
13
14     NodeList nodeList = document.getElementsByTagName("book");
15
16     for (int i = 0; i < nodeList.getLength(); i++) {
17         Node node = nodeList.item(i);
18         if (node.getNodeType() == Node.ELEMENT_NODE) {
```

Output - xmlproject (run) x

```
run:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Year: 1925
Genre: Fiction
-----
Title: To Kill a Mockingbird
Author: Harper Lee
Year: 1960
Genre: Fiction
-----
Title: 1984
Author: George Orwell
Year: 1949
Genre: Dystopian
-----
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Modifying XML Data



The screenshot displays an IDE interface with the following components:

- Top Bar:** Includes icons for file operations and a memory usage indicator showing 318.9/470.0MB.
- Project Explorer (Left):** Shows a project structure with folders 'books', 'JavaApplication1', and 'xmlproject'. Under 'xmlproject', there are files 'XmlParser.java', 'Xmlproject.java', and 'books.xml'.
- Members View (Bottom Left):** Shows the 'main' method of the 'XmlParser' class.
- Editor (Right):** Displays the 'XmlParser.java' file with the following code:

```
31 System.out.println("Author: " + author);
32 System.out.println("Year: " + year);
33 System.out.println("Genre: " + genre);
34 System.out.println("-----");
35 }
36
37
38 // Modify the year of the first book
39 Element firstBook = (Element) nodeList.item(index: 0);
40 firstBook.getElementsByTagName(name: "year").item(index: 0).setTextContent(textContent: "2023");
41
42 // Save the modified XML to a new file
43 TransformerFactory transformerFactory = TransformerFactory.newInstance();
44 Transformer transformer = transformerFactory.newTransformer();
45 DOMSource source = new DOMSource(n: document);
46 StreamResult result = new StreamResult(new File(pathname: "updated_books.xml"));
47 transformer.transform(xmlSource: source, outputTarget: result);
48
49 } catch (Exception e) {
50     e.printStackTrace();
51 }
52
53 }
54
```

```
// Modify the year of the first book

Element firstBook = (Element) nodeList.item(0);
firstBook.getElementsByTagName("year").item(0).setTextContent("2023");


// Save the modified document to a new file

TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();

DOMSource source = new DOMSource(document);

StreamResult result = new StreamResult(new File("updated_books.xml"));
transformer.transform(source, result);


System.out.println("XML file has been updated and saved as updated_books.xml.");
}

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



## Output

The screenshot shows an IDE with the following components:

- Projects View:** Shows a project named 'xmlproject' with source files 'XmlParser.java', 'Xmlproject.java', 'books.xml', and 'updated\_books.xml'.
- Source Editor:** Displays the code in 'XmlParser.java'. The code parses an XML document, finds the first book element, and updates its year to '2023'. It then saves the modified XML to a new file 'updated\_books.xml'.
- Navigator:** Shows the 'main' method of the 'XmlParser' class.
- Output Console:** Displays the output of the program, showing the details of three books: 'The Great Gatsby', 'To Kill a Mockingbird', and '1984'. The output ends with 'BUILD SUCCESSFUL (total time: 0 seconds)'.

```
39 Element firstBook = (Element) nodeList.item(index: 0);
40 firstBook.getElementsByTagName(name: "year").item(index: 0).setTextContent(textContent: "2023");
41
42 // Save the modified XML to a new file
43 TransformerFactory transformerFactory = TransformerFactory.newInstance();
44 Transformer transformer = transformerFactory.newTransformer();
45 DOMSource source = new DOMSource(n: document);
46 StreamResult result = new StreamResult(new File(pathname: "updated_books.xml"));
47 transformer.transform(xmlSource: source, outputTarget: result);
48
49 } catch (Exception e) {
```

run:

Title: The Great Gatsby  
Author: F. Scott Fitzgerald  
Year: 1925  
Genre: Fiction  
-----  
Title: To Kill a Mockingbird  
Author: Harper Lee  
Year: 1960  
Genre: Fiction  
-----  
Title: 1984  
Author: George Orwell  
Year: 1949  
Genre: Dystopian  
-----  
BUILD SUCCESSFUL (total time: 0 seconds)

## updated\_books.xml

The screenshot shows an IDE with the following components:

- Toolbar:** Includes icons for file operations (new, open, save, etc.), a configuration dropdown set to "<default config>", a memory usage indicator showing 285.0/563.0MB, and several status icons.
- Project Explorer (Left):**
  - books
  - JavaApplication1
  - xmlproject
    - Source Packages
      - xmlproject
        - XmlParser.java
        - Xmlproject.java
        - books.xml
        - updated\_books.xml (selected)
    - Test Packages
    - Libraries
    - Test Libraries
- Source Editor (Right):** Displays the content of updated\_books.xml:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?><library>
2   <book>
3     <title>The Great Gatsby</title>
4     <author>F. Scott Fitzgerald</author>
5     <year>2023</year>
6     <genre>Fiction</genre>
7   </book>
8   <book>
9     <title>To Kill a Mockingbird</title>
10    <author>Harper Lee</author>
11    <year>1960</year>
12    <genre>Fiction</genre>
13  </book>
14  <book>
15    <title>1984</title>
16    <author>George Orwell</author>
17    <year>1949</year>
18    <genre>Dystopian</genre>
19  </book>
20 </library>
```
- Navigator (Bottom Left):** Shows the XML tree structure:
  - <? version="1.0" encoding="UTF-8" standalone="no"?>
  - library
    - book
    - book
    - book