1. Definición del contexto de la aplicación

La aplicación desarrollada se centra en procesar flujos de datos en formato RSS (Really Simple Syndication), que es un estándar basado en XML utilizado para distribuir contenido actualizado como noticias o blogs. El propósito principal es extraer artículos relacionados con "inteligencia artificial" desde un feed RSS proporcionado por *Europa Press*. Posteriormente, la aplicación almacena los datos obtenidos en un archivo XML estructurado, lo que facilita su reutilización o análisis.

¿Qué es XML y RSS?

- **XML (Extensible Markup Language):** Es un lenguaje de marcado diseñado para almacenar y transportar datos de manera estructurada, facilitando la interoperabilidad entre sistemas. En este proyecto, se utiliza para estructurar los datos extraídos del RSS.
- **RSS:** Es un formato basado en XML que permite a los usuarios recibir actualizaciones de contenido de manera automática. Contiene elementos estándar como `<title>`, `<link>`, `<pubDate>`, entre otros, que describen artículos o elementos del feed.

2. Librerías y herramientas utilizadas

El proyecto se implementó en Java, utilizando las siguientes librerías estándar:

- **`javax.xml.parsers` y `org.w3c.dom`:** Para la lectura y creación de documentos XML.
- **`java.net.HttpURLConnection`:** Para realizar conexiones HTTP y recuperar el contenido del feed RSS.
- **`javax.xml.transform`:** Para transformar y escribir el contenido procesado en un archivo XML.

Estas herramientas permiten gestionar los datos en XML y realizar operaciones de red de manera eficiente.

3. Proceso de desarrollo y planificación

Planificación inicial

1. **Definición de requisitos:**
    - Leer datos desde un feed RSS.

- Procesar y extraer elementos clave como título, fecha de publicación, autor y enlace.
- Generar un archivo XML con los datos estructurados.

2. **Diseño de componentes principales:**
   - Clase `Noticia`: Representa un artículo con atributos como título, autor, fecha y enlace.
   - Clase `RssParser`: Se encarga de conectarse al feed RSS, leer y extraer los datos.
   - Clase `XmlWriter`: Crea un archivo XML a partir de la lista de noticias.
   - Clase `Main`: Coordina las operaciones principales y ejecuta el flujo del programa.

Fases del desarrollo

- **Fase 1:** Implementación de la clase `Noticia` para modelar los datos.

```java
public class Noticia {
    private String titulo;
    private String fechaPub;
    private String autor;
    private String link;


    public Noticia() {
    }

    public Noticia(String titulo, String fechaPub, String autor, String link) {
    }

    // Getters y setters
    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getFechaPub() {
        return fechaPub;
    }

    public void setFechaPub(String fechaPub) {
        this.fechaPub = fechaPub;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public String getLink() {
        return link;
    }

    public void setLink(String link) {
        this.link = link;
    }
}
```

● **Fase 2:** Desarrollo del `RssParser` para extraer información del feed RSS utilizando DOM (Document Object Model).

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
```

```java
public class RssParser {
    public List<Noticia> parseRss(String rssUrl) throws Exception {
        List<Noticia> noticias = new ArrayList<>();

        // Realizar la conexión HTTP para recuperar el RSS
        URL url = new URL(rssUrl);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");

        try (InputStream inputStream = connection.getInputStream()) {
            // Parsear el XML
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            Document document = builder.parse(inputStream);

            // Recupera del documento la lista de elementos <noticia>
            NodeList items = document.getElementsByTagName("item");
            //Recorremos la lista de <noticia> guardando los datos de cada una en l
            for (int i = 0; i < items.getLength(); i++) {
                Element noticiaElement = (Element) items.item(i);
                //Se guarda cada elemento como una nueva noticia con titulo, fecha
                Noticia noticia = new Noticia();
                noticia.setTitulo(getTagValue("title", noticiaElement));
                noticia.setFechaPub(getTagValue("pubDate", noticiaElement));
                noticia.setAutor(getTagValue("author", noticiaElement));
                noticia.setLink(getTagValue("link", noticiaElement));

                noticias.add(noticia);
            }
        }
        //Devuelve la nueva lista de noticias parseada
        return noticias;
    }
```

```java
//Método que recupera el elemento con la etiqueta seleccionada
private String getTagValue(String tagName, Element element) {
    NodeList nodeList = element.getElementsByTagName(tagName);
    if (nodeList.getLength() > 0) {
        return nodeList.item(0).getTextContent();
    }
    return null;
}
```

- **Fase 3:** Creación del `XmlWriter` para estructurar los datos y generar el archivo XML.

```java
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;
import java.io.File;
import java.util.List;

public class XmlWriter {
    /*
    * Metodo que recibe la lista de noticias del rss y lo escribe en el nuevo documento xml
    *Parametros:
    * noticias La lista de noticias recuperada del rss
    * fileName nombre del archivo en el que se escribiran los datos
    * */
    public void writeRssItemsToXml(List<Noticia> noticias, String fileName) throws Exception {

        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.newDocument();

        // Crear el elemento raíz
        Element root = document.createElement("noticias");
        document.appendChild(root);

        // Añadir elementos
        for (Noticia noticia : noticias) {
            Element nodoNoticia = document.createElement("noticia");

            appendChildWithText(document, nodoNoticia, "titulo", noticia.getTitulo());
            appendChildWithText(document, nodoNoticia, "fechaPub", noticia.getFechaPub());
            appendChildWithText(document, nodoNoticia, "autor", noticia.getAutor());
            appendChildWithText(document, nodoNoticia, "link", noticia.getLink());

            root.appendChild(nodoNoticia);
        }
```

```
// Escribir el archivo
TransformerFactory transformerFactory = TransformerFactory.newInstance();
Transformer transformer = transformerFactory.newTransformer();
// Configurar UTF-8 y sangrías
transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8"); // Configuración de UTF-8
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2"); // Conf
DOMSource source = new DOMSource(document);
StreamResult result = new StreamResult(new File(fileName));
transformer.transform(source, result);
```

```
/*
Metodo que añade un nodo al xml
Parametros:
    document Documento al que se añade el nodo
    parent El nodo padre al que se añada el nuevo nodo
    tagName El nombre del nodo a añadir
    textContent el texto del nodo
*/
private void appendChildWithText(Document document, Element parent, String tagName, String textContent) {
    Element child = document.createElement(tagName);
    child.setTextContent(textContent);
    parent.appendChild(child);
}
```

- **Fase 4:** Integración de todas las partes en la clase `Main` y pruebas iniciales con un feed RSS real.

```java
import java.io.PrintStream;
import java.nio.charset.StandardCharsets;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        // Configurar la consola para UTF-8
        try {
            System.setOut(new PrintStream(System.out, true, StandardCharsets.UTF_8.name()));
        } catch (Exception e) {
            System.err.println("No se pudo configurar UTF-8 para la consola: " + e.getMessage());
        }

        String rssUrl = "https://www.europapress.es/rss/rss.aspx?buscar=inteligencia-artificial";

        try {
            // Parsear el RSS
            RssParser parser = new RssParser();
            List<Noticia> noticias = parser.parseRss(rssUrl);

            // Mostrar los datos en consola
            System.out.println("Artículos encontrados:");
            for (Noticia item : noticias) {
                System.out.println(item);
            }

            // Generar archivo XML
            String outputFileName = "25-11-2024-list.xml";
            XmlWriter xmlWriter = new XmlWriter();
            xmlWriter.writeRssItemsToXml(noticias, outputFileName);

            System.out.println("\nEl archivo XML ha sido creado: " + outputFileName);
        } catch (Exception e) {
            System.err.println("Error al procesar el RSS: " + e.getMessage());
        }
    }
}
```

Dificultades encontradas

- Validación del feed RSS y manejo de posibles errores en la conexión HTTP. Incorporamos manejo de errores mediante bloques try-catch para capturar excepciones relacionadas con la conexión HTTP o el formato del feed.
- Estructuración del archivo XML para que sea comprensible y legible. Para solucionarlo, configuramos el transformador XML para que formatee el archivo con sangrías y espacios, mejorando su legibilidad.
- Encontrar una de las librerías adecuadas y aprender a manejarla para realizar el proyecto. Las resolvimos buscando en la documentación. Nos apoyamos en la documentación oficial de las librerías seleccionadas (javax.xml.parsers, org.w3c.dom, javax.xml.transform) para entender su funcionamiento y aplicarlas en el proyecto. Exploramos ejemplos y guías en línea que explicaban cómo utilizarlas para parsing y generación de XML.

4. Ejecución del proyecto

Requisitos previos

- Tener Java JDK instalado (versión 8 o superior).
- Un entorno de desarrollo como IntelliJ IDEA o terminal con herramientas de compilación.

Instrucciones para ejecutar

- Abrir el proyecto en el IDE.
- Compilar y ejecutar.

Formato del archivo XML generado
Esto generará un archivo llamado 25-11-2024-list.xml en el directorio de trabajo. El archivo XML contiene los datos de las noticias con la siguiente estructura:

```
<noticias>

    <noticia>
        <title>Título del artículo</title>
        <pubDate>Fecha de publicación</pubDate>
        <author>Autor</author>
        <link>URL del artículo</link>
    </noticia>
    ...
</noticias>
```

5. Pruebas realizadas

Pruebas funcionales

- **Conexión al feed RSS:** Verificación de la recuperación correcta de datos desde `https://www.europapress.es/rss/rss.aspx?buscar=inteligencia-artificial`.
- **Extracción de datos:** Confirmación de que los elementos `<title>`, `<pubDate>`, `<author>` y `<link>` son procesados adecuadamente.
- **Generación de XML:** Validación de que el archivo XML contiene todas las noticias extraídas.

Casos de prueba

1. **Feed válido con noticias:**
    - Resultado esperado: Archivo XML con todas las noticias extraídas.
2. **Feed vacío o inaccesible:**
    - Resultado esperado: Un mensaje de error informativo en la consola.
3. **Manejo de caracteres especiales:**
    - Verificación de que caracteres como `&` o `<` no interrumpen la generación del XML.

Resultados

Todos los casos de prueba se ejecutaron correctamente, y el archivo XML fue generado sin errores.

Conclusión

El proyecto cumple con los objetivos planteados, permitiendo procesar datos de un feed RSS y almacenarlos en un formato XML reutilizable. La modularidad del diseño facilita la ampliación de funcionalidades, como agregar más campos de las noticias o procesar múltiples feeds RSS.

Anexos

- Código fuente completo.

  https://github.com/Alexfh94/EjercicioXML/tree/master/src

- Ejemplo de archivo XML generado.

  https://github.com/Alexfh94/EjercicioXML/blob/master/25-11-2024-list.xml