XML Converter Overview

Contents

Introduction

The CSV, TXT, XLSX to XML Converter is a powerful tool designed to simplify the process of converting data from various file formats, including CSV (Comma-Separated Values), TXT (Plain Text), and XLSX (Microsoft Excel), into structured XML (Extensible Markup Language) files. This tool enables users to effortlessly transform and store their data in XML format, providing compatibility, flexibility, and ease of integration with a wide range of applications and systems.

This document provides an understanding of the key components and interactions within the converter, outlining its design principles and functionality.

Architecture Overview

This document is the first approach to present the information of this project in a structured fashion and discuss its architecture.

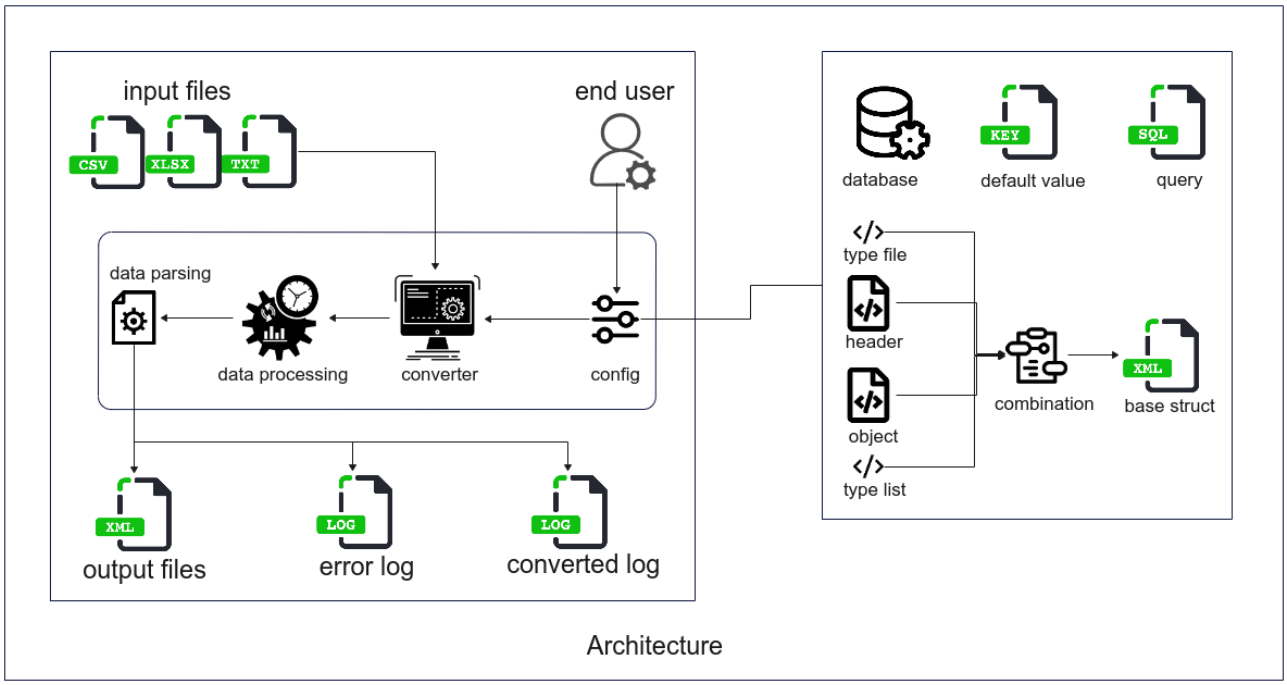
Whenever possible, we make use of existing technology instead of reinventing the wheel and the usability of the system is taken into account as the #1 priority.

The structure that the rest of the document will follow is:

1. A summarized description of the software architecture, including major components and their interactions.

2. The technologies and libraries chosen.

Architecture Summary



Architecture

1. Configure base struct: The end user will configure the necessary components for the data conversion process. This includes configuring the database connection, default values, queries, XML header, and XML object files.

2. Build base struct: Based on the configurations from the previous step, you will create a base struct. This base struct will serve as the fundamental structure for handling the data conversion process.

3. Input files into the converter: The end user will provide input files to be converted.

4. Process data: The converter will read data from the input files and initiate the data parsing process based on the built base struct. This process may involve extracting information from the input files, converting data formats, executing database queries, or any other data conversion-related tasks.

5. Generate output files: Once the data conversion process is successfully completed, the converter will generate new output files. These files will contain the successfully parsed data from the input files.

6. Log errors: If any errors occur during the conversion process, the converter will log error information into an error log file. This file will include details about the input files that failed to convert and provide specific error descriptions.

7. Provide converted files: Lastly, the converter will provide the successfully converted files to the end user. This allows the end user to review the results and utilize the parsed data successfully for further purposes.

Technologies and libraries

1. Language and Technologies:
    - Language: Java (version JDK 19)
    - GUI Framework: Java Swing
    - Database: Oracle DB
    - XML Processing: XML APIs available in Java
2. Additional Libraries:
    - in4j: A library for handling input and output operations.
    - jasypt: A library for encryption and decryption of sensitive information.
    - ojdbc8: A JDBC driver for connecting to the Oracle database.
    - univocity parsers: A library for parsing and processing CSV, TSV, and other text-based formats.
3. Source Code Repository:
    - The source code for the conversion tool is stored on GitHub, allowing for version control, collaboration, and easy access to the codebase.
4. Development and Deployment:
    - The tool is developed using Java programming language and Java Swing for the graphical interface.
    - The required libraries and dependencies (in4j, jasypt, ojdbc8, univocity parsers) are included in the project configuration.
    - The source code is version-controlled using Git and stored on GitHub for easy access, collaboration, and tracking of changes.
    - The tool can be compiled into an executable JAR file or packaged as an installer for easy deployment on different systems.

Predefined Convention

The CSV, TXT, XLSX to XML Converter follows a set of predefined conventions to ensure consistency and standardization in the conversion process. These conventions include:

System	
Delimiter	If the delimiter of the input files is known in advance, it is necessary to specify the delimiter. Failure to specify the delimiter may result in the tool automatically detecting the delimiter, which can lead to potential errors during the conversion process.
Log	
Error	The error log is used to display any errors that occur during the conversion process.
Date	The date log is used to display information about the input files that have been successfully converted and their corresponding output files for each day. Each log file will have a name in the format "yyyymmdd.log," representing the date of conversion.
Path folder	
Input	Configure the path for the folder containing the input files to expedite the file selection process. By default, it is set to the path "log/input".
Output	Configure the path for the folder to store the successfully converted output files. By default, it is set to the path "log/output".
Error	Configure the path for the folder to store the input files that encountered errors during the conversion process. By default, it is set to the path "log/error".
Process	Configure the path for the folder to store the input files that have been successfully processed and converted. By default, it is set to the path "log/process".
See path	Display the paths of the specified folders. It allows users to view the configured directory paths for various folders.

Connect to DB	
Database connection	If the conversion process requires using data from a database, it is necessary to establish a database connection before performing the conversion. The configuration information is provided by the administrator. ***Please note that the administrator should only provide accounts that have read-only access to the database and are not permitted to make any changes to the data.***
Default value	
Name	Name must be contain only letters, numbers, and underscore.
Value	Accessing a value based on its specified name.
Query	
Name	Name must be contain only letters, numbers, and underscore.
Query	Using a query to retrieve data from a database, if the query result returns multiple columns and/or multiple rows, it will fetch the data from the first row of the first column only.
Template (XML)	
Tag	Name must be start with a letter and not contain special characters or spaces except underscore.
Attributes	• Ref: Used to remove an entire parent tag if at least one column name inside the ref does not exist in the input files' header or if the value in that column in the input files does not match the declared value in the ref. • Use: Used to specify whether a tag is mandatory and must have data to exist or not. • Type: Used to validate whether the data matches the expected format or not. • Format: Used to format the data of a tag, such as removing any excess or unnecessary parts.
Value	• From File: The data will be retrieved from the input files using the pattern ${COLUMN_NAME} where COLUMN_NAME represents the specific column name. • From DB: The data will be retrieved from the

	database using the query that has been previously configured in the "Query" section, using the pattern @{QUERY_NAME} where QUERY_NAME represents the specific query name. • From Default Values: The data will be retrieved from the predefined values configured in the "Default value" section, using the pattern *{DEFAULT_VALUE_NAME} where DEFAULT_VALUE_NAME represents the specific default value name. • From Generator: The data will be automatically generated by the tool using the specified generation type, using the pattern #{GENERATE_TYPE} where GENERATE_TYPE represents the specific type of generation. • From Template: The data will be static and predefined from a template, meaning it will not change during the conversion process.
Template name	Name must be contain only letters, numbers, underscore and unique.
Header (Template)	
Tag	Same as template's convention.
Attributes	Same as template's convention but header cannot use “ref” attribute.
Value	Same as template's convention but header cannot use value “From file”.
Header name	Same as template's convention.
Object (Template)	
Tag	Same as template's convention.
Attributes	Same as template's convention.
Value	Same as template's convention.
Header name	Same as template's convention.
Struct	
Type file	It must be start with letter and contain only letters, numbers and underscore.
Header	Choose from list headers.

Type list	It must be start with letter and contain only letters, numbers and underscore.
Object	Choose from list objects.
File name output	• From Struct: The data will be static and predefined from a struct, meaning it will not change during the conversion process. • From DB: The data will be retrieved from the database using the query that has been previously configured in the "Query" section, using the pattern @{QUERY_NAME} where QUERY_NAME represents the specific query name. • From Default Values: The data will be retrieved from the predefined values configured in the "Default value" section, using the pattern *{DEFAULT_VALUE_NAME} where DEFAULT_VALUE_NAME represents the specific default value name. • From Generator: The data will be automatically generated by the tool using the specified generation type, using the pattern #{GENERATE_TYPE} where GENERATE_TYPE represents the specific type of generation.
Struct name	It must be contain only letters, numbers and underscore.
Input files	
Support	.csv (CSV), .xlsx (Excel), .txt (Text)
Data	The data in the input files should follow a specific format where the first row represents the column names, and the subsequent rows contain the corresponding data.
Date format	ISO 8601 format is recommended: yyyy-MM-dd HH:mm:ss

Setup Guide

- Click download JDK and install according to JDK Installation Guide (Require install JDK 19 or higher)

- After installing Java, you can simply double-click on the "xml-converter.jar" or "xml-converter.exe" file to launch the tool. It will automatically generate the necessary resources for the system, and you can use it as usual.