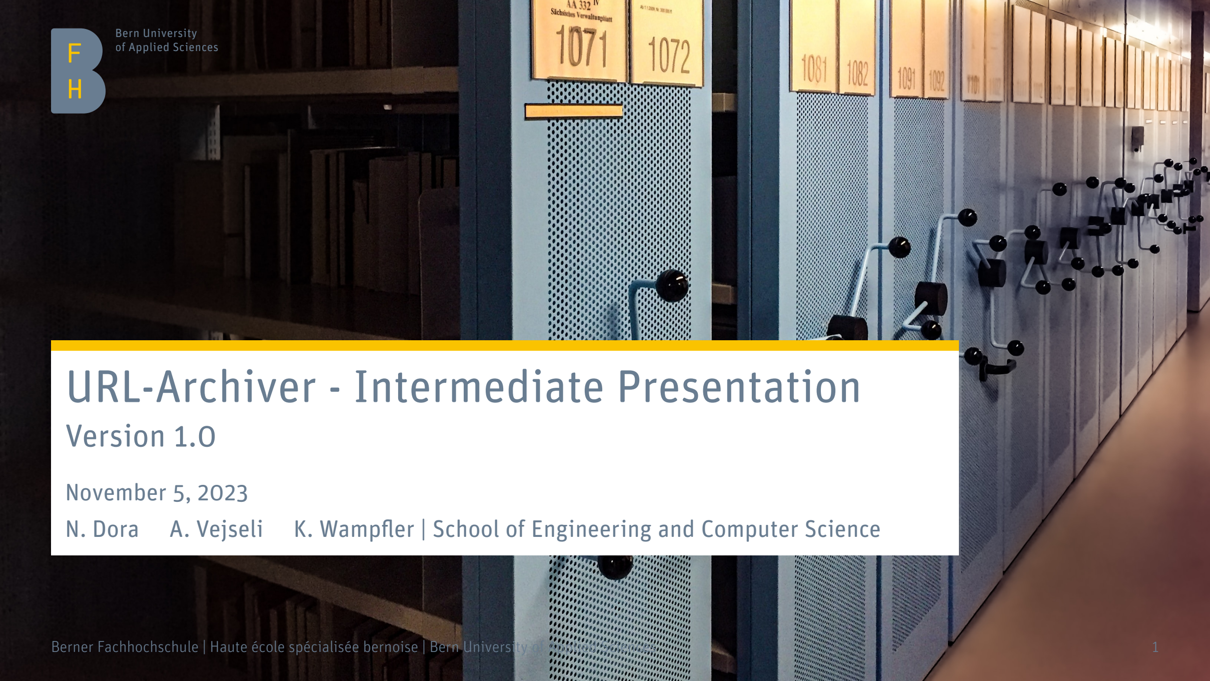


Bern University of Applied Sciences

URL-Archiver - Intermediate Presentation

Version 1.0

November 5, 2023

N. Dora A. Vejseli K. Wampfler | School of Engineering and Computer Science

Table of Content

Topics Covered

Problem Statement

Problem Statement

The ever changing internet

- ever-changing internet (websites are taken down or changed)
- not a problem for everyday internet users but…
- …what about documents (e. g. theses, documentations etc.)
    - invalid links to content-relevant information
    - invalid quote links
    - old documents may contain numerous broken hyperlinks

Problem Statement

Simple Solution

- Archive the actual state of the linked websites
  - Wayback machine
  - archive.today
  - Memento Time Travel
  - and many more…
- But who has the time and motivation?
  - …to search each hyperlink
  - …to manually archive each website

Objectives

- Develop a software that…
  - …searches hyperlinks within documents
  - …is capable of archiving linked websites
  - …can store current and archived links in a file
  - …is easy and intuitive to use

Requirements

- The software must be…
  - …a Java application that is compatible with multiple platforms
  - …FLOSS-licensed
  - …capable of archiving websites on archive.ph or/and WayBackMachine
  - …capable of generating a CSV-file with key-value (URL, archived URL)

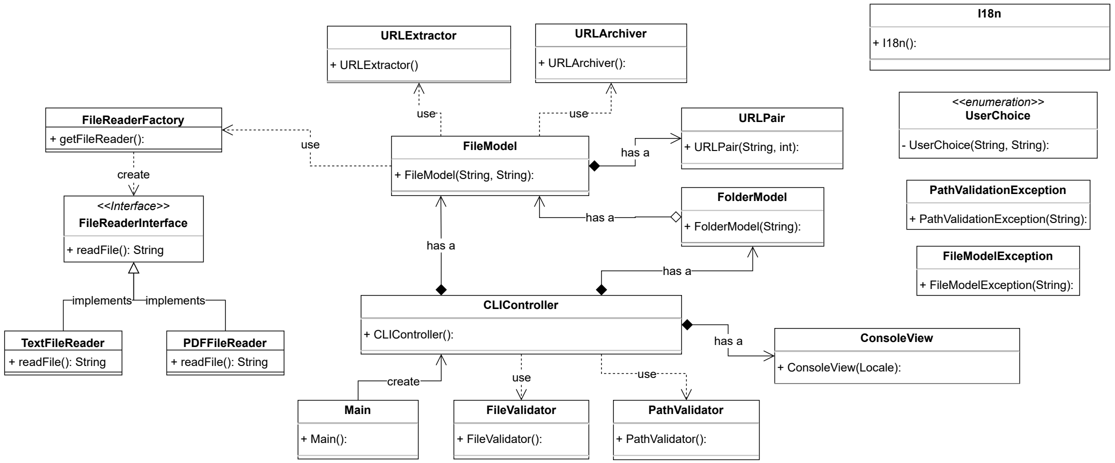
General conditions

- The program code should be minimal, modular, and self-explaining
- The program code should be published in a git repository
- The project must be carried out according to scrum
- Public documents should be written in English
- LaTeX should be used for the project report

Solving The Problem

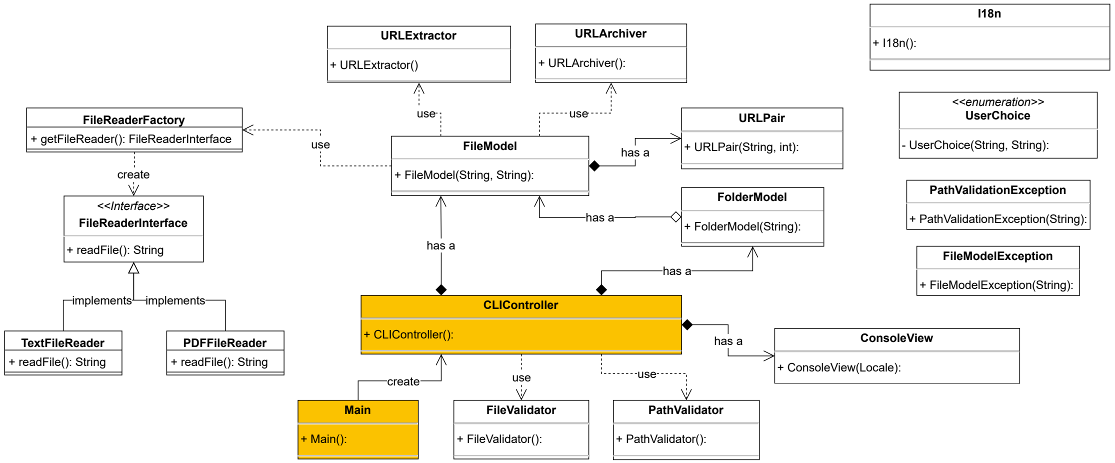
High Level Class Diagram

Overview



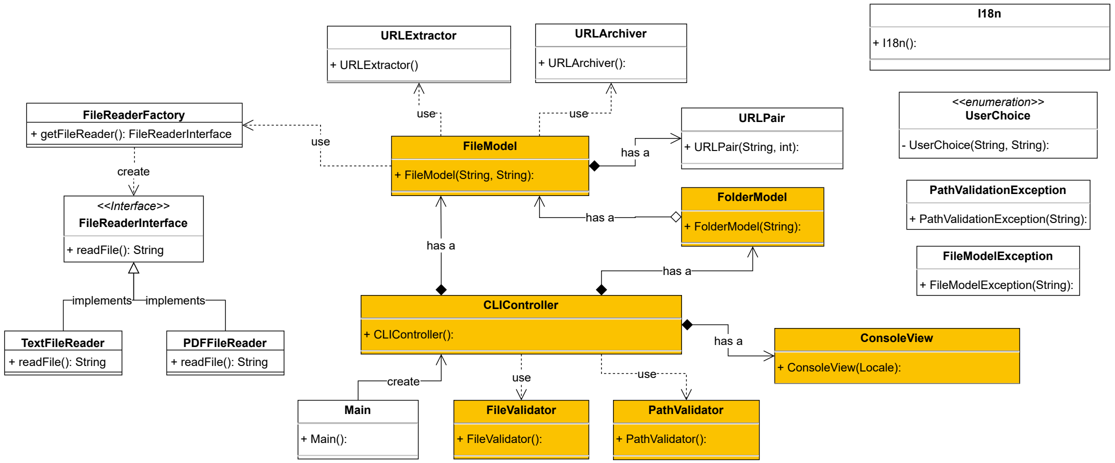
High Level Class Diagram

Main



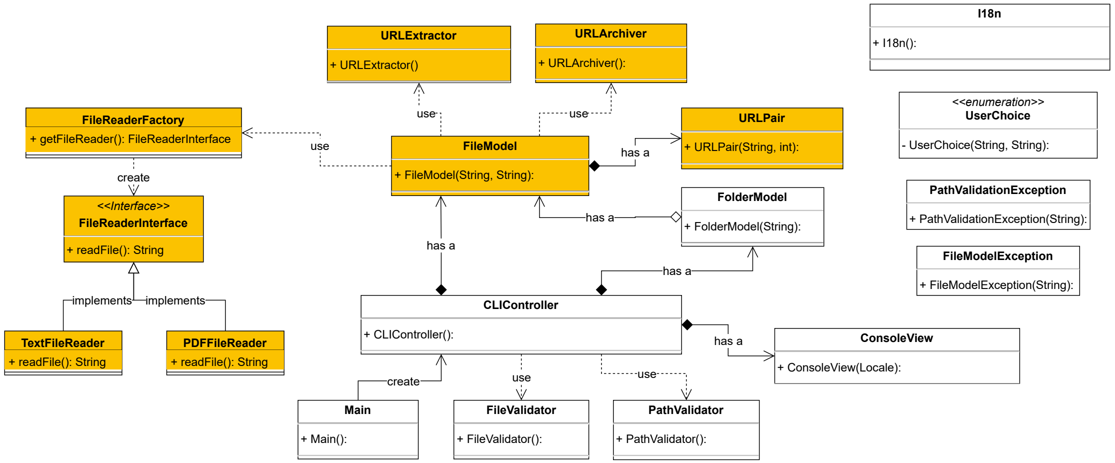
High Level Class Diagram

CLIController



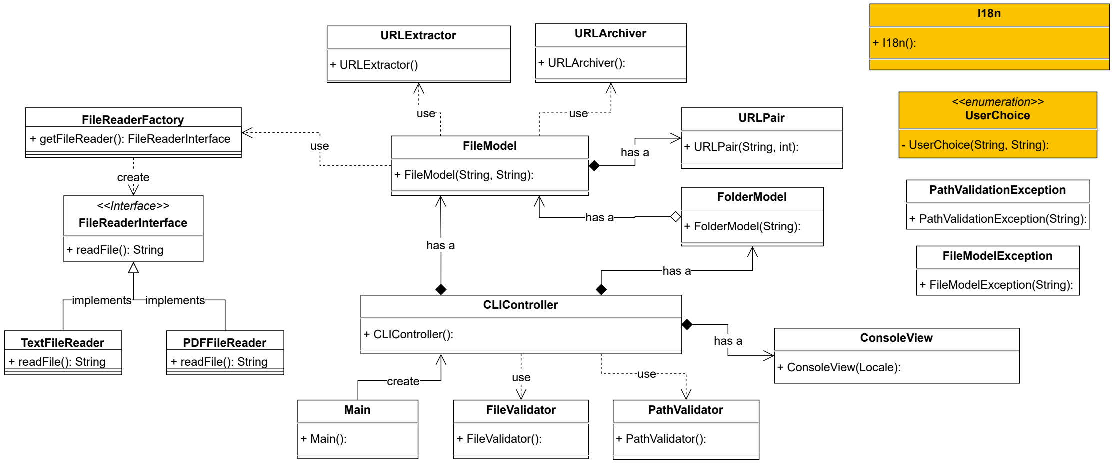
High Level Class Diagram

FileModel



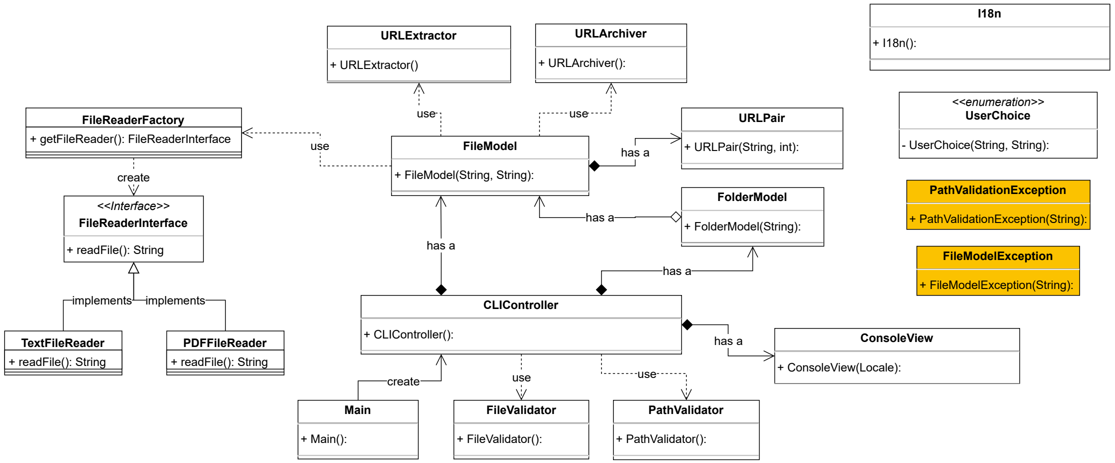
High Level Class Diagram

I18n and UserChoice



High Level Class Diagram

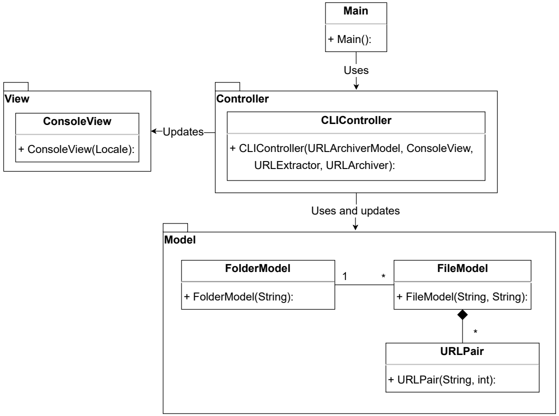
Exceptions



Architecture

Employing the MVC Pattern

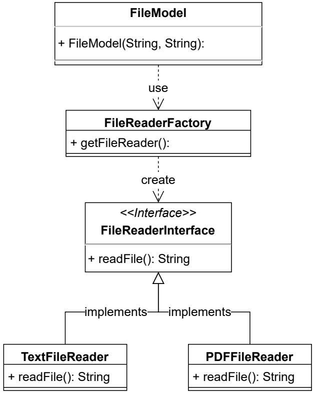
- Modular design for easy extension.
- Separate data, interface, and control flow.
- Facilitates the potential addition of a GUI.



Architecture

SOLID Principles

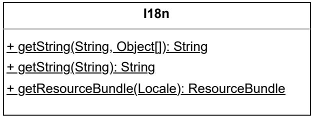
- Scalable and robust design.
- Ensures maintainability and clear code structure.
- Commitment to best programming practices.



Architecture

Multilanguage Support with ResourceBundle

- Ready for global adaptability.
- Future-proofing architectural choice.
- Potential to accommodate multiple languages.

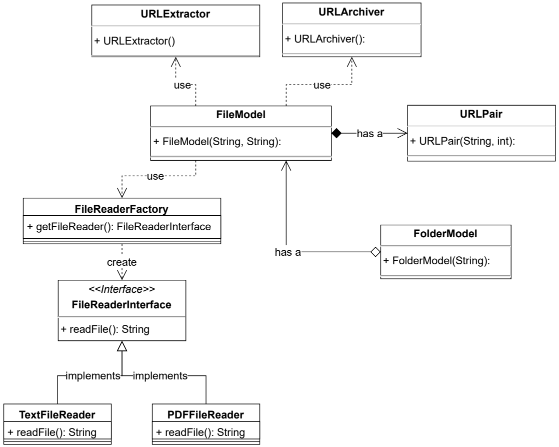

I18n
+ getString(String, Object[]): String
+ getString(String): String
+ getResourceBundle(Locale): ResourceBundle

Overview of the Data Model

The data model is designed to represent and manage the data structures essential for the application’s operation. It encompasses handling files, user interactions, and archiving URLs.

Key Components of the Data Model

- **FileModel**: Represents and handles individual files.
- **FolderModel**: Manages a collection of FileModel instances.
- **URLPair**: Links extracted URLs with their archived counterparts.
- **URLArchiverModel**: Organizes and maintains URLPairs.
- **UserChoice**: Enumerates possible user commands and actions.



Interactions and Relationships

This model’s components interact to facilitate file and URL management:

- FolderModel aggregates FileModels representing files in a folder.
- URLArchiverModel processes and pairs URLs from FileModels.
- User actions guide the flow of data through the models.

Use Case Flow in the Data Model

A typical use case might involve:

1. A user selects an action (UserChoice).
2. The system processes files in a folder (FolderModel with FileModels).
3. URLs are extracted, archived, and paired (URLArchiverModel with URLPairs).

This flow demonstrates the integrated operation of the data model components.

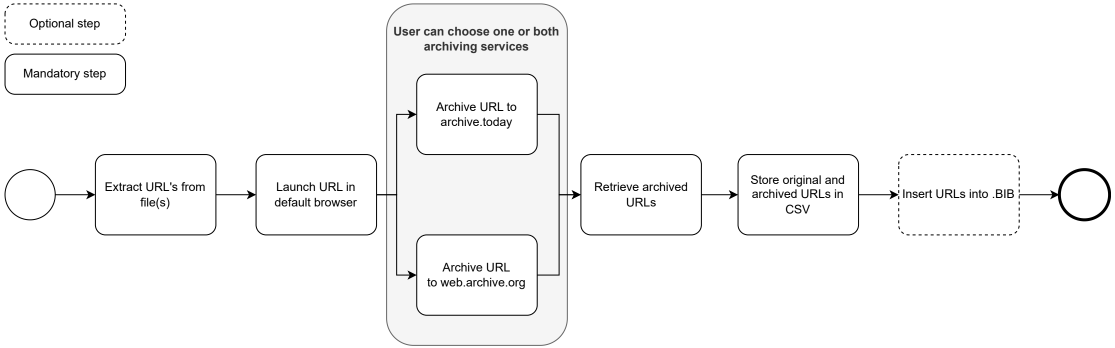
Conclusion

The data model is a critical framework within the application, enabling structured data manipulation, storage, and retrieval, which are fundamental to the application’s functionality and user experience.

Process model

Workflow for URL Extraction and Archiving

- User can skip URLs, launch them, access help or quit the program at any time.


Optional step
Mandatory step
Extract URL's from file(s)
Launch URL in default browser
User can choose one or both archiving services
Archive URL to archive.today
Archive URL to web.archive.org
Retrieve archived URLs
Store original and archived URLs in CSV
Insert URLs into .BIB

Technologies I

Core Technologies

- **Java**: The primary programming language for our application.
- **LaTeX**: Used for documentation and presentation.

Supporting Technologies

- **JUnit 5**: Utilized for unit testing.
- **PDFTextStripper (PDFBox library)**: Used for extracting text from PDF documents.
- **Selenium**: Web automation tool used for tasks like inputting URLs, handling captchas, and retrieving archived URLs due to the absence of an official API from archive.today.
- **Maven**: Essential for compilation, dependency management, and building the project.

Technologies II

Archiving Services

- **Wayback Machine**: One of the services used by the URL-Archiver to archive URL’s.
- **Archive.today**: One of the services used by the URL-Archiver to archive URL’s.

etc.

Project Management with SCRUM

Scrum-Rollen

Sprintziele

Anforderungen

Scrum Adaptionen

etc.

Blocks

Block with a title

Content.

Without title

Block types

Exampleblock

Content.

Alertblock

Content.

Example (Example environment)

Content.

section pages