

INTLIB

ETL Framework for RDF Data and its Application to Legislation Documents (SW Project Proposal)

Supervisor: Tomáš Knap (tomas.knap@mff.cuni.cz)

Team members: 5 students

Language: Java

OS: Windows 7 / Windows Server 2008 / Linux

Goal of the Project

The goal of the project is to build an ETL (Extract-Transform-Load) tool for RDF data. Such tool will support creation of data processing units (DPUs) - extractors, transformers, and loaders - and definition of data processing pipelines consisting of the desired DPUs. The application will have graphical user interface for administration of the ETL process, such as creation of data processing pipelines, monitoring and debugging pipelines' execution.

Such tool will be applied to the domain of legislation documents to extract important aspects of the legislation documents, convert them to RDF data format, transform them accordingly and load them to RDF database. DPUs needed for such demonstration will be implemented. Such use case will be presented during the defense as the main use case of the project.

The tool will use experience obtained while developing SW project ODCleanStore (CLEAN) successfully defended in December 2012.

Impact

The ETL tool developed should become part of the LOD2 stack, a stack of tools creating as part of the European project LOD2 (<http://lod2.eu>).

Important Functional Requirements

- User may define multiple data processing pipelines with components for fetching and manipulation of RDF data. User may edit/delete such pipelines.
- User may list his pipelines and pipelines being shared with him.
- User may run/debug the pipelines.
- User may schedule the pipeline to run at certain periods of time or when another pipeline finishes.
- User may browse the RDF data produced by the pipeline, the RDF data being in database.
- User is notified via email whether his scheduled pipeline was executed or whether there was an error in the pipeline, in which case the error is described in detail. A possibility to be notified via other means, such as data feeds, should be also examined.
- User may view the result of the pipeline execution, view errors.
- User may share the pipeline with others.
- User may define permissions for the pipeline, which users may use/edit the pipeline.
- The system will support the following DPUs:
 - Extractors: Fetching data from SPARQL endpoint, RDF file/directory of files (at least RDF/XML, TTL and TRIG serializations), extracting XML files according to XSLT, extracting data from CSV files (if enough time).

- Loaders: Loading data to RDF file (at least RDF/XML, TTL, TRIG serializations), Loading data to RDF database (SPARQL endpoint). Loader may support the data with metadata (when created, by who, which DPUs were executed on the way).
- Transformers: Transformer being able to execute SPARQL queries, Linker (Silk).
- DPUs needed by the legislation documents use case (e.g., extractors from text based on GATE, deduplication of obligations and rights, etc.).
- User may create new DPUs, copy the existing DPUs.
- User may share the DPU with others (already preconfigured or not configured at all).
- User may import/export DPUs.
- Administrator may import/export configuration of the whole system.
- Administrator may manage the whole DPU tree.
- Administrator may specify certain well-known SPARQL endpoints, e.g. , staging database the pipelines are running on top, knowledge base containing ontologies being used by DPUs and quarantine – database which may be used to store data which needs further manual cleansing
- The system will validate the processed RDF data (at least syntactically).
- The system will use ontologies in the knowledge base to suggest the ontology terms when configuring DPUs (must be supported by the particular DPU).
- The system will support management of namespace prefixes
- Administrator can create new users and delete existing ones, change passwords; the system will provide standard forgotten password functionality.
- The system will log users' activities in detail, logging can be configured (the target of the logs: database, file; the granularity of the logs). Timestamps will be associated with the log records.
- The system will provide statistics – which pipelines ran for how long, under which user account, what was the size of the result, resulting state - error/OK, average run time for each pipeline, number of pipelines per user account, total runtime per user account – all of this computed from a comprehensive log.

Important use cases are depicted in Figure 12.

Graphical user interface

The application will involve graphical administration user interface enabling to:

- List and define new DPUs - extractors, transformers, loaders.
 - Figure 8 depicts list of DPUs and detail for each DPU.
 - Figure 9 shows creation of new DPU.
- List the data processing pipelines available (Figure 6).
- Define new data processing pipeline (which utilizes certain extractors, transformers, loaders).
 - Figure 1 depicts the basic information about pipeline and specification of permissions.
 - Figure 2 depicts the tree of DPUs available to be used by the data processing pipelines.
 - Figure 3 depicts the canvas on which the pipeline may be defined by drag&dropping DPUs from the tree of DPUs (Figure 2).
 - Figure 4 depicts how the pipeline may be debugged while it is created.
 - All Figures 1 – 4 are reachable easily from one screen using hide-able panels.
- Monitor results of the processing pipelines (Figure 5).

- Browse the data (view on the RDF triples in the resulting data, SPARQL querying interface to query the resulting data). External tools may be used for realizing the RDF views or providing SPARQL query interface.
- Management of scheduled rules (Figure 11).
- Management of users and roles (Figure 10) , management of locked pipelines, pruning records in the monitor, define new SPARQL endpoints, settings of email notifications.
- Depict statistics about the executed pipelines, running times of pipelines etc.
- Main menu should be as depicted in Figure 7 (minor modifications may be needed)

Non-functional Requirements

- Documentation must be in English. The interface for creation of new DPUs must be well described – the document must be simple, intuitive, easy to read for DPU providers.
- The system must support processing of big RDF dumps (GBs).
- OSGi framework is used for loading custom DPUs.
- Project is maven based and will be hosted on GitHub.
- Representative web page about the tool must be created and maintained during the project duration.
- Usability and intuitiveness of user interface is crucial.
- Repeatable unit and integration tests and also testing data (GBs) + pipelines (10s) will be prepared by the team.
- The team is managed by using some ticketing system with the possibility to track number of hours done.
- Language: Java
- Supported OS: Windows 7 / Windows Server 2008 / Linux
- Coding style - All classes, non-private methods and attributes must have meaningful English description
- Iterative development

Expected Utilization of the Team

(Including analysis, documentation, and testing of the introduced parts):

- Application's core business logic, communication of GUI components with the core logic, communication of scheduler with the core logic, storage configuration, permissions, OSGi framework for custom DPUs, architecture of custom DPUs (2.0 persons)
- Graphical User Interface (1.7 persons)
- Particular DPUs needed, applying ETL tool to legislation documents (1 person)
- Internal team management – organizing meetings, notes from the meetings, planning milestones, checking tasks done (0.3 person)

Expected Work Plan:

We will deliver the project iteratively; every iteration involves unit, component, and integration testing and documentation draft, so that the results are usable immediately. The work plan is as follows, assuming deadline and the final version of the project in Month 9:

- general analysis, specification, architecture; specification of the components, selection of the features for the first iteration (Months 1- 2)
- Iteration 1 – a possibility to define a pipeline, use one or two DPUs, execute the pipeline; no permissions, no debugging, no monitoring of pipeline executions, no DPUs management (Month 3)
- Further month iteration (Month 4 – 7): Iterations 2-5 + continuous testing and documentation
- Final testing, bug solving, minor improvements, documentation – user, programmer, configuration guide, installer (Months 8 - 9)

Figures

Name:

Description:

Permissions:

User	Read (Copy,Run)	Developer	Actions
Everyone	<input type="checkbox"/>	<input type="checkbox"/>	
martin	<input checked="" type="checkbox"/>	<input type="checkbox"/>	delete
(type for new perm)			

Callout 1 (left): Line with "Everyone" permission settings is always present as a first line in the table.

Callout 2 (top right): This panel can be hidden to the top of the screen. Name remains visible, Useful when user needs more space for editing pipeline.

Figure 1. Pipeline Description & Permissions

Filter:

- [-] Extractors
 - ▶ Extractor from RDF file
 - ▼ Extractor from XML file
 - My Extractor from XML file
 - ▶ Extractor from CSV file
 - ▶ Extractor from SPARQL endpoint
 - ▶ BEs extractor from SPARQL endpoint
 - ▶ Extractor from TXT act expression
 - ▶ Extractor from CSV file with act metadata
- [+] Transformers
 - ▶ SILK Linker
 - ▶ Object Replacer
 - ▶ Predicate Replacer
 - ▶ SILK BEs Deduplicator
 - ▶ Deduplicator of defined law terms
- [+] Loaders
 - ▶ Loader to RDF file
 - ▶ Loader to XML file
 - ▶ Loader to SPARQL endpoint
 - ▶ Criminality reports
 - ▶ Legislation documents

Callout 1 (left): This panel (DPU hierarchy) can be hidden to the left of the screen. Useful when user needs more space for editing pipeline.

Callout 2 (middle): Certain elements of the hierarchy, such as whether it is Extractor, Transformer or Loader may be expressed differently than just by subfolders. Regarding other folders:

- two categories (generic, configured) + tags, DPUs hidden to the user will not be shown
- more categories as folders, DPU may be in more categories, no tags, DPUs hidden to the user will not be shown

Figure 2. Tree of DPUs – Extractors, Transformers, Loaders

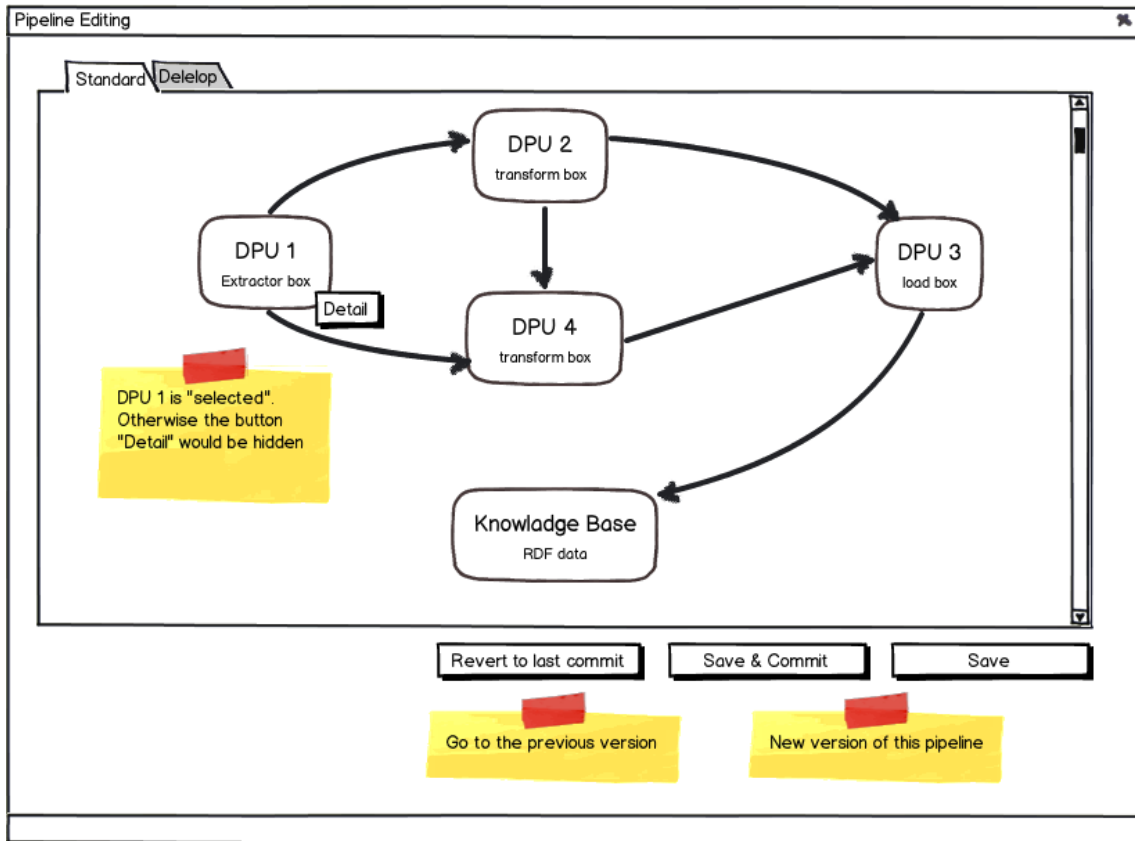


Figure 3. Pipeline Canvas

Debug / Test Pipeline

Time	Type	Source	Short Message	
2013.02.13	⚠	SPARQL Extractor	Can't Connect to SPARQL endpoint	show details
2013.03.04	✓	Extractor from XML file	Finished OK	show details

Log with error messages coming from the different DPUs

Graph

SPARQL Query:

```
SELECT *
WHERE {
  ?subject ?predicate ?object.
}
```

Browse staging database view
this is just a example of what it could look like:
simple SPARQL query interface

Query

subject	predicate	object

Figure 4. Pipeline Debugging

The screenshot shows two main components of the Pipeline Debugging interface:

Filters Panel: Contains a search bar, a 'Running' dropdown menu, and a 'Delete filters' button. Below is a table of pipeline runs:

Date	Name	User	Progress	Debug	Finished no errors	Report
2012.02.12	Ext	Petr				
2012.02.13	Alf	Petr				2 errors
2012.12.12	Ext	Petr				debug data

Buttons for 'Stop', 'Show log', and 'debug data' are visible. At the bottom, 'Page size: 10' and 'Page: 1 2' are shown.

Messages Overview Panel: Shows 'Messages overview for: Pipeline: Ext' with 'Start: 01.09.2012' and 'End: 01.09.2012'. It includes 'User: Petr'. A table lists messages:

Time	Typ	Source	Short message
2012.02.13		RdfSource	Can't connect to the show

'Close' and 'Export' buttons are at the bottom. A bracket below the panels is labeled 'Side window'.

Figure 5. Pipeline Execution Monitoring

The screenshot shows the 'Workspace home' interface for pipeline execution monitoring:

Pipelines List: A table with columns 'ID', 'Name', 'Author', and 'Actions'. A 'Filter' dropdown is set to 'type in...'.

ID	Name	Author	Actions
1	ted	tomasknap	detail copy run debug delete schedule
2	isvzus	tomasknap	detail copy run debug delete schedule
3	lex	martin	

A 'Create Pipeline' button is located at the bottom right of the list.

Yellow Note: A yellow sticky note with a red tab contains the text: 'Debug -> Like run, but the intermediate results (the content of the graphs filled during the pipeline execution is stored and can be observed later by the user) Debug flag in the monitor screen'.

Figure 6. List of Pipelines

The screenshot shows a navigation bar with the following items: 'Pipelines', 'DPUs', 'Execution (Monitor)', 'Browse Data', 'Scheduler', 'Settings', 'Administrator', 'Help', and 'Logout'.

Figure 7. Main Menu

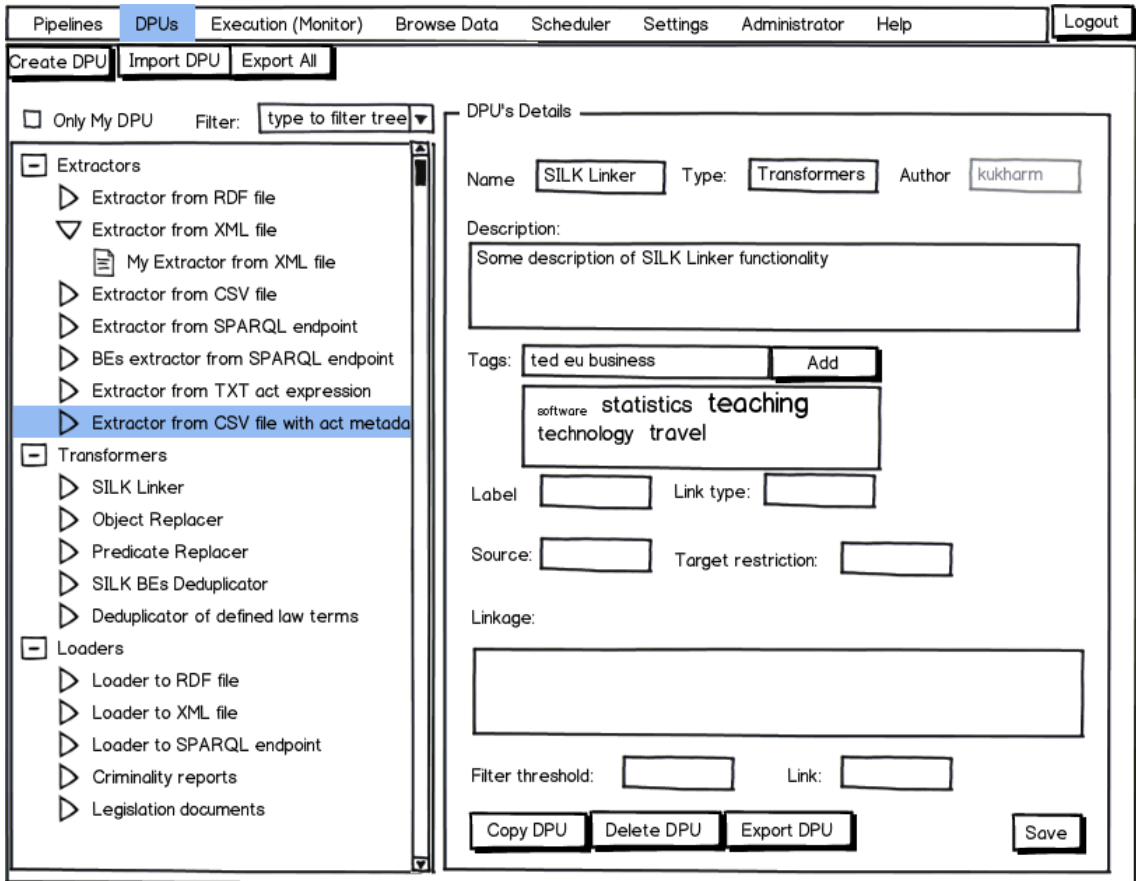


Figure 8. List of DPU's and DPU's Detail

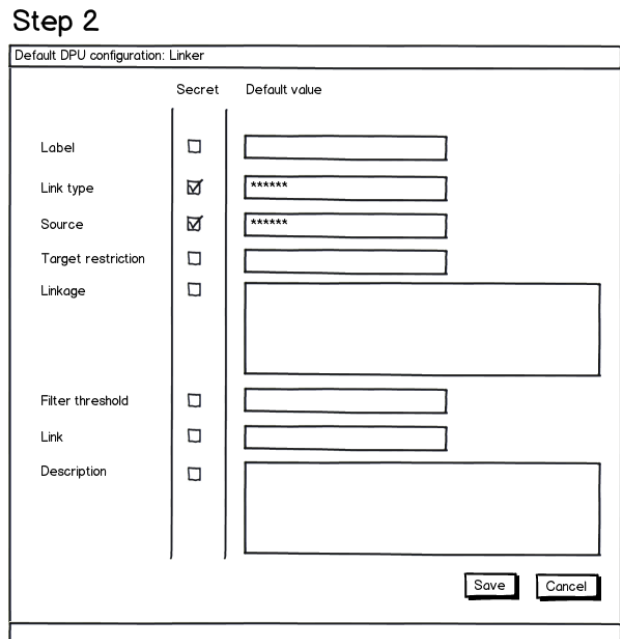
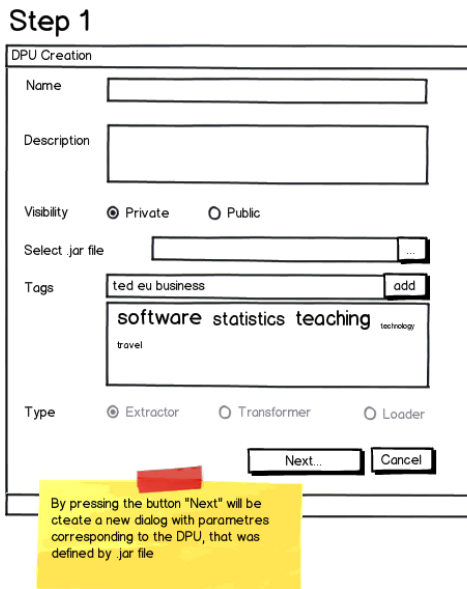


Figure 11. Scheduling

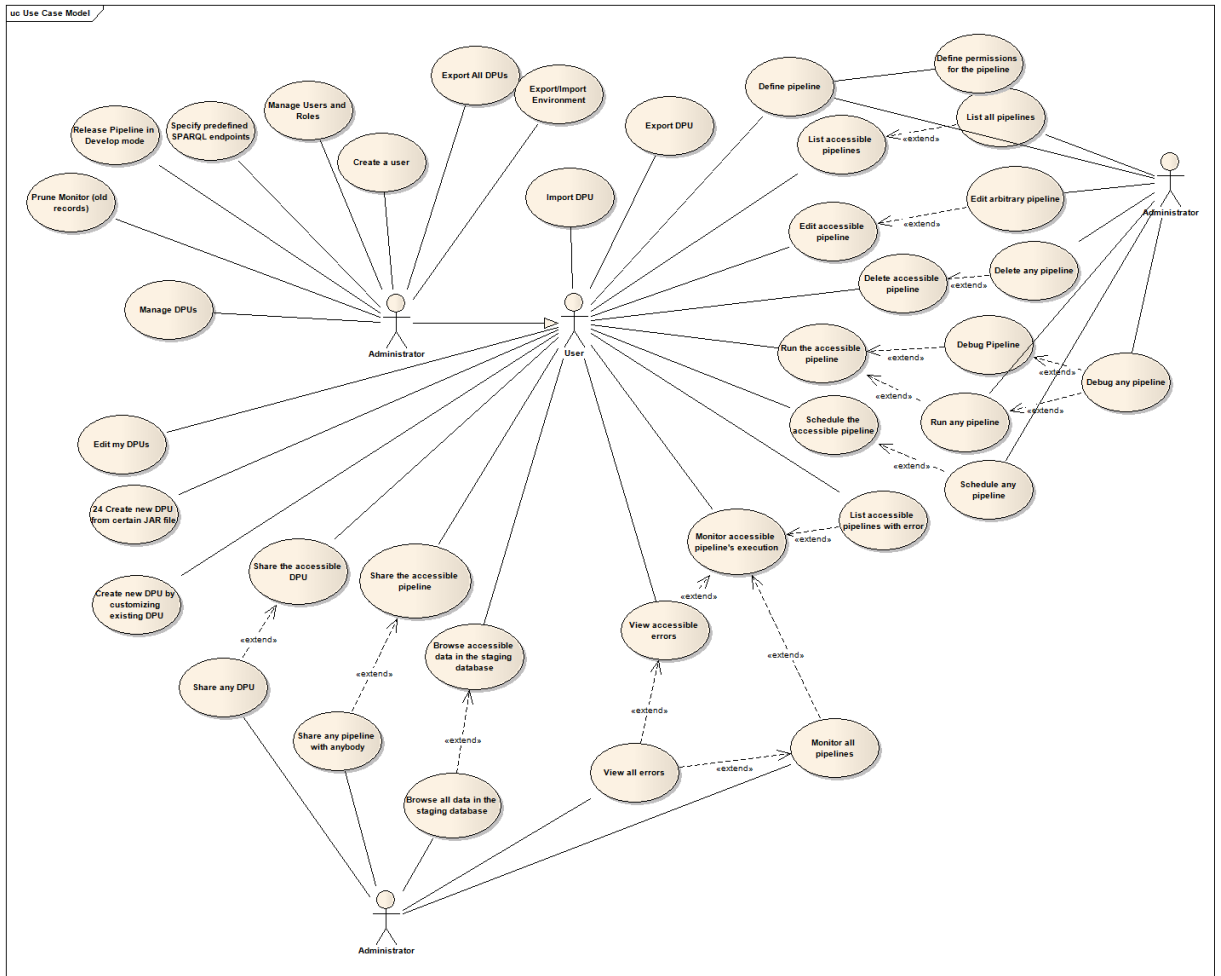


Figure 12. Use Cases