TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Andreas Huber

SLR Toolkit
Informal Documentation

Tasks: Metadata and LaTex export

Outline

1. Essentials and tasks
2. Metadata Plugin
3. LaTex Export Plugin
4. Possible Enhancements
5. Development – general pointers and getting started
6. Contact

Essentials and Tasks

Essentials and Tasks

- SLR Toolkit - A Toolkit for Systematic Literature Reviews (https://github.com/sebastiangoetz/slr-toolkit/)

- Developed as an Eclipse RCP Application

- Tasks
  - ✓ *"Provide a way to add metadata to an SLR project"*
  - ✓ *"Provide a view with key statistics of the project"*
  - ✓ *"Enable the export of half-filled Latex Papers"*

Metadata Plugin

Metadata Plugin

SLR Toolkit

File Search Window LaTex Export Mendeley

Bibtex Entries | Project Explorer

Projekt

type filter text

/Bachelorarbeit_taxonomie/Library.bib

Problems | Properties

0 items

Description	Resource

meta.slrproject

Key facts

File: C:\Uni\Dev\Bachelorarbeit_Kevin_Horst\meta.slrproject

Title: Title of the systematic literature review

Keywords: keyword1, keyword2, keyword3, keyword4, keyword5, keyword6, keyword7, keyword8, keyword9

Key statistics

Refresh — Push "Refresh" to display the current key statistics.

Number of main dimensions: 8

Number of dimensions: 128

Number of documents: 58

Authors

Add | Edit | Delete

John Doe; johnd@mail.tld; University of XYZ
Name; Email; Affiliation

Description

Abstract: Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad

Description of the taxonomy: Lorem ipsum dolor sit amet, consectetur adipisici elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

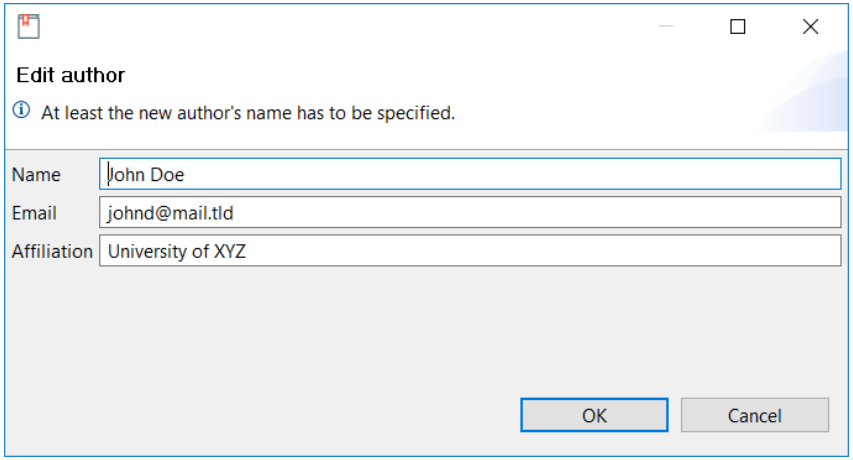
ChartView

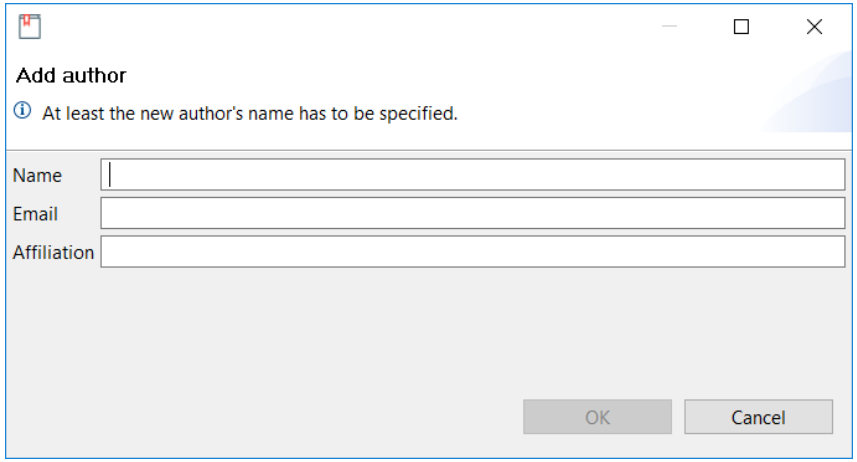
There is no Data to display at the moment.
Try clicking a Term with subclasses.

Taxonomy

- Zielgruppe (0)
- Veroeffentlichungsland (0)
- Veroeffentlichungssort (0)
- Art (0)
- Thema (0)
- Anwendungsfeld (0)
- Ergebnis (0)
- Methodik (0)

Metadata Plugin





Metadata Plugin

- *"Provide a way to add metadata to an SLR project"*
  - Relevant metadata are **title** of the project, the **authors** and **keywords**, an **abstract** and a **description of the taxonomy**
  - Derived tasks
    - Representation of the metadata
    - Persistency and storage
    - Viewing and editing

- *"Provide a view with key statistics of the project"*
  - Key statistics are the number of dimensions and documents
  - Derived tasks
    - Get current key statistics
    - Keep the view in sync with the key statistics

- Developed plugin: **de.tudresden.slr.metainformation**

Metadata Plugin

General workflow

- **MetainformationEditor** view instantiation is triggered through opening of an .slrproject file
- Since the taxonomy and documents are needed for the key statistics, the project's files containing these have to be initialized in the editing domain, see *init()* and *initializeWholeProject()*
- The view has a **DataProvider**, which encapsulates information for the key statistics
- Adding and editing an author trigger **NewAndEditAuthorDialog** instantiation, see *addListeners()* in *MetainformationEditor*
- Editing data in the view marks the editor as dirty, saving stores the current data in the file which is the source of the metainformation, see *doSave()*
- Case: multiple projects are „opened"
  - Editing domain can hold the information of only one project
  - Setting focus to the metainformation view triggers unload and reinitialization of the whole project, see *setFocus()*

Metadata Plugin

Representation of the metadata / Persistency and storage

- Package *de.tudresden.slr.metainformation.data*
  - Both classes **SlrProjectMetainformation** and **Author** are POJOS representing the project's metadata and a list of the authors
- Data is stored in an XML file in the project
  - Realized through JAXB, annotated POJO classes
  - Unmarshalling in class **MetainformationUtil**, marshalling and writing to file in **MetainformationEditor** in method *doSave()*

Metadata Plugin

Viewing and editing

- Package *de.tudresden.slr.metainformation.editors*
- **MetainformationEditor** is the view which shows the metadata and provides a way in form of a form to manipulate the data
- The slide about the general workflow already contains most of the relevant information, have a look at *init()* and *createPartControl()* to see how the view is initialized and the GUI is created
- There is a **ControlListener** which listens to resize events to ensure that the scrollbars are working correctly

Metadata Plugin

Get current key statistics / keep the view in sync with them

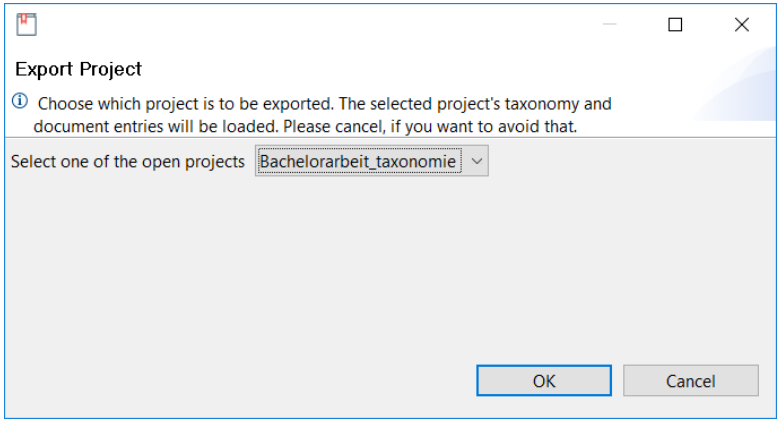
- **DataProvider** feeds the current key statistics
- There is a refresh button to get the current key statistics
- The actual objective was to implement a listener, which signals that either the taxonomy or the documents have changed and to let that trigger a refresh of the key statistics
- Due to permission issues in Eclipse RCP (Listeners to views from SWT elements) and the implementation of both data sources this was not possible, that is why there is a workaround via a button

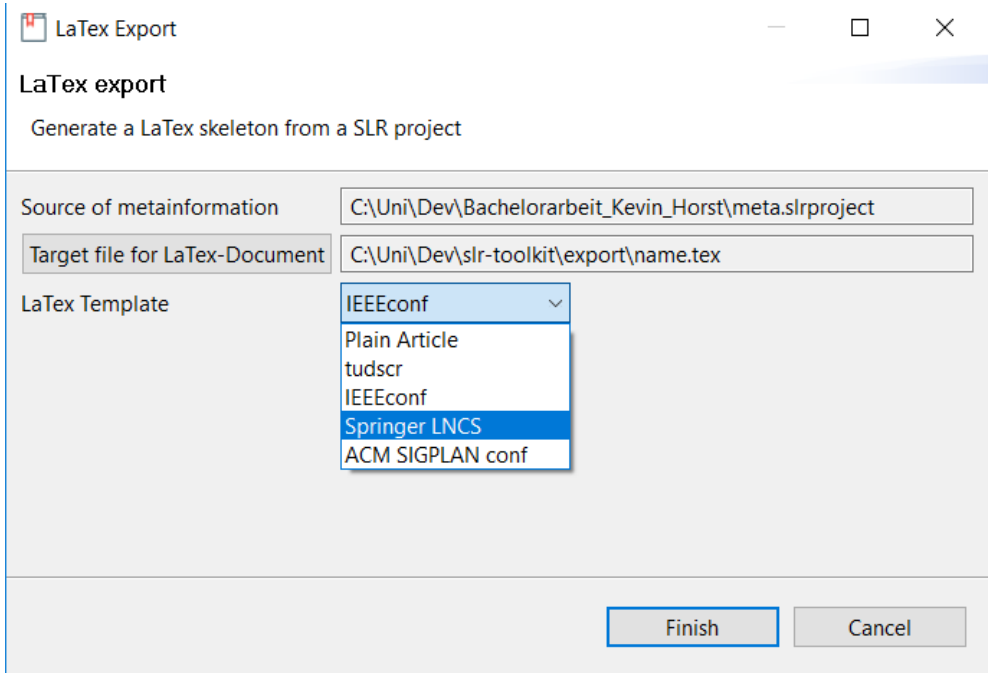
TECHNISCHE UNIVERSITÄT DRESDEN

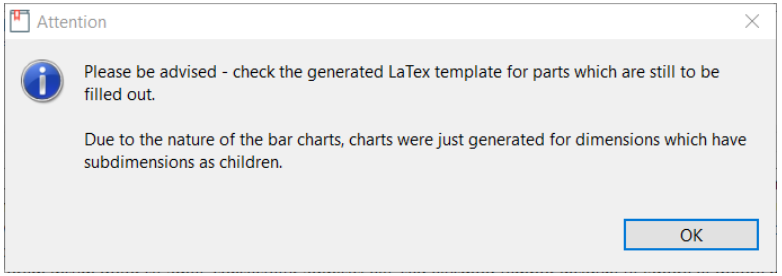
DRESDEN concept

LaTex Export Plugin

LaTex Export Plugin







LaTex Export Plugin

- *"Enable the export of half-filled Latex papers"*
  - Derived tasks
    - Collection of data – solved through **DataProvider** and **SlrProjectMetainformation** classes from the metainformation plugin
    - Different templates and filling of papers
    - Export of charts

- Developed plugin: **de.tudresden.slr.latexexport**

LaTex Export Plugin

General workflow

- **exportHandler** is called via menu bar
- If less than two projects are in the workspace, the selection of the project to be exported is trivial, otherwise a **ExportProjectChooser** dialog opens and prompts the selection of a project, see *tryLoadingProjectFiles()* in exportHandler
  - for the only available or selected project, the documents, taxonomy and metadata are initialized in the editing domain and metainformation plugin
- A **LatexExportWizard** is opened, asking to specify a path to the export document and for a selection of a template
- Export generation is triggered by *performExport()* in LatexExportWizard
  - Therefore, a **SlrLatexGenerator** is generated which specifies the template and the content
- Charts are generated and exported to an „images“ folder in the export path
- The document template is filled and exported

LaTex Export Plugin

Different templates and filling of papers

- Every template has an own class, see package *de.tudresden.slr.latexexport.latexgeneration.documentclasses*
- The .tex templates (with slots for the input, slot names are in class **SlrLatexTemplate**) and auxiliary files are located in the „resources“ folder of the plugin
- SlrLatexTemplate is the abstract superclass for all templates
  - Concrete templates have to specify URLs to the resources (hardcoded in constructor)
  - Due to differences in displaying the author, abstract,… certain generator functions can or have to be overridden, e.g. *generateAuthorSection(), fillDocument()*
  - *fillDocument()* is kind of an template method for the insertion of content
- The content of the template in the resources folder is read into a String (*getResourceContentAsString()* in **SlrLatexGenerator)**, the slots from the template are filled using the **StrSubstitutor** from Apache Commons and paths to the images (Mapping from Term to path as String) are inserted
- The resulting String is written to a .tex file

LaTex Export Plugin

Adding a new template (1/2)

- To add a new template, information has to be hardcoded into the plugin
- In class **SlrLatexTemplate**:
  - Add a static final string containing the name of the template, this name will be displayed in the export wizard
  - Add this string to the array *documentTypes*
- add the new file of the template in a new folder in the path *resources/latexTemplates/*
  - at least one .tex file has to be added, auxiliary files are optional
- Generate a new class for the template in package *de.tudresden.slr.latexexport.latexgeneration.documentclasses* extending **SlrLatexTemplate**
  - The constructor has to specify the path to the template (templatePath), URLs to every resource and an array containing all resources

```
public TemplateIEEEconf() throws MalformedURLException {
    URL resource0 = new URL(resourcePrefix + "latexTemplates/ieee/IEEEtran.cls");

    this.filesToCopy = new URL[] { resource0 };
    this.name = "IEEEconf";
    this.templatePath = new URL(resourcePrefix + "latexTemplates/ieee/ieee-template.tex");
}
```

LaTex Export Plugin

Adding a new template (2/2)

- In the constructor of **SlrLatexGenerator**, add an if-statement to check whether the selection matches an existing template. If so, set *concreteDocumentTemplate* to an instance of the new document class

```
if (templateName.equals(SlrLatexTemplate.TEMPLATE_PLAIN))
    conreteDocumentTemplate = new TemplatePlainArticle();
if (templateName.equals(SlrLatexTemplate.TEMPLATE_ACM))
    conreteDocumentTemplate = new TemplateACMSigplanConf();
if (templateName.equals(SlrLatexTemplate.TEMPLATE_IEEE))
    conreteDocumentTemplate = new TemplateIEEEconf();
if (templateName.equals(SlrLatexTemplate.TEMPLATE_SPRINGER_LNCS))
    conreteDocumentTemplate = new TemplateSpringerLncs();
if (templateName.equals(SlrLatexTemplate.TEMPLATE_TUDSCR))
    conreteDocumentTemplate = new TemplateTUDSCR();
```

- That‘s it!

LaTex Export Plugin

Export of charts

- In class **LatexExportGenerator**, the charts for the main dimensions are generated, using the Eclipse BIRT charting engine
- In the only method of the class, a mapping is returned which maps a Term to a String, which represents a path where the chart is put. The path is defined relatively, e.g. Term „a“ is mapped to „images/a.JPG“
- The charts are then located in the defined output folder for the export in an images folder
- Originally, the format of the output should be in svg- or pdf-format, but due to unfixed bugs in BIRT, it‘s just possible to export to JPG or other non-vector graphics formats

Possible Enhancements

Possible Enhancements

- Replace BIRT with another charting engine that supports vector graphics output
- Refactor template system to minimize hardcoded information for the templates
- Refactor the metainformation view
- Find a way to sync key statistics and the view without having to push the refresh button

Development

Development

The very beginning

- The learning curve for the SLR Toolkit and Eclipse RCP in general is very steep, at some point, there will probably be frustration, but it‘s worth to keep digging
- Some very fine tutorials can be found at http://www.vogella.com/tutorials/eclipse.html

Development

SLR Toolkit, practical hints for the beginning

- Either use the **DataProvider** from the metainformation plugin to get the **current** dimensions and documents
- Or get it via
  - SearchUtils.*getDocumentList()*
  - ModelRegistryPlugin.*getModelRegistry().getActiveTaxonomy()*
- when developing via Eclipse, try to keep your dependencies and required/imported packages correct to ensure that when trying to build via Maven there are not that many errors

Contact

- Feel free to contact me any time if there are questions: andreas@andhub.de