ElanModules

User's Guide

Tom Ruette

ElanModules: User's Guide

by Tom Ruette

Version ${project.version}

Table of Contents

List of Tables

Foreword

This document is a guide to the user of how to use the ELAN pepper module and how to utilize a mapping performed by it. Also, this document provides a closer look at the details of such a mapping in a declarative way, to give the user a chance to understand how specific data will be mapped by the ELAN pepper module.

Chapter 1. Overview

This project contains the pepper modules listed in Table 1.1, “pepper modules contained in this project”. A single module can be identified via its coordinates (module-name, format-name, format-version) also given in Table 1.1, “pepper modules contained in this project”. You can use these coordinates in a pepper workflow description file to identify the modules in a pepper conversion process. A description of how to model a workflow description file can be found under https://korpling.german.hu-berlin.de/saltnpepper/.

Table 1.1. pepper modules contained in this project

Name of pepper module	Type of pepper module	Format (if module is im- or exporter)
ElanImporter	importer	eaf

Chapter 2. ElanImporter

General information about this importer.

Mapping to Salt

For the moment, only Elan files without a linked media file and that only contain text-based annotations are properly handled. Moreover, the module has been developed within the framework of one specific project (Deutsch Diachron Digital), which may lead to some unexpected behavior when exposed to alien files. Nonetheless, the development of the mapping has been oriented towards generic functionality. In case of problems, please contact the developer.

The first step in the mapping to Salt consists of the creation of a so called `STextualDS`, which contains the complete text that is going to be annotated. The tier in Elan that holds the text is given by the user in the special properties.

The second step in the mapping to Salt consists of creating so-called `STokens`. These `STokens` are calculated on the basis of the tier in the list of segmentation-tiers that has the most amount of annotations (Note that this might be an assumption that does not hold for your files). The other tier(s) in the segmentation layer parameter is used to create (a) segmentation layer(s) that contains Salt Spans, which are given `SOrderRelations` (if wanted).

The actual mapping of the Elan annotations to Salt Annotations takes the simple form of a loop in a loop. For every tier in Elan, that tier is considered from left to right (i.e. from beginning to the end). For every encountered Elan annotation on the tier, a corresponding `SSpan` on the segmentation layer is searched. If an `SSpan` is found with an identical time slot, the annotation for the current tier is added to the SSpan. If the Elan annotation has no corresponding SSpan, it might be the case that the Elan annotation is a span of SSpans. If that is the case, the Elan annotation is added to this newly created span of `SSpan`. If the Elan annotation is shorter than all available `SSpan`, then this might be a subtoken, below the segmentation layer. Since we have all the `STokens` available, we can simply search for the (group of) `STokens`(s) that match the length of the Elan Annotation.

Properties

The table Table 2.1, “properties to customize importer behaviour” contains an overview of all usable properties to customize the behavior of this pepper module. The following section contains a close description to each single property and describes the resulting differences in the mapping to the salt model.

Table 2.1. properties to customize importer behaviour

Name of property	Type of property	optional/ mandatory	default value
elan.importer.primTextTierName	textual value	mandatory	
elan.importer.segTierNames	textual value	mandatory	
elan.importer.ignoreTierNames	textual value	optional	
elan.importer.addSOrderRelation	Boolean	optional	true

elan.importer.primTextTierName

Name of the tier containing the primary text. If the primary text is split out over multiple segments in Elan, then the values of all segments will be concatenated, including whitespace. The concatenated string is set as `STextualDS` in Salt. Only a single tiername is allowed in this property.

elan.importer.segTierNames

Names of the tiers that will be used as segmentation layers. It is possible to only set a single segmentation layer. Segmentation layers are useful when there are subtoken annotations (i.e. annotations of parts in the `STextualDS` that are not typically considered as tokens), because it allows the end-user to select which segmentation he or she wants to see. Multiple tiernames are allowed in this property, and tiernames should be separated by commas.

elan.importer.ignoreTierNames

Names of the tiers that will be ignored. Sometimes, specific Elan tiers are not needed in another format (e.g. annotator comments), and this parameter allows you to get rid of them. Obviously, these tiernames can not be in the previous two properties. Multiple tiernames are supported in this property, and tiernames should be separated by means of commas.

elan.importer.addSOrderRelation

Determines if, this module shall add `SOrderRelations` to the segmentation layer(s). If only one segmentation layer is provided, no order relations are needed, and they will not be set. This property is default set to true, because it does not make much sense to have this to false.