Import Overview

Classes and important methods are in **bold**, events are in *italics*.

TaskController::postAction

saves the **task (project) entity** to the database

creates a **DataProvider**, depending from where the data comes (Zip, SingleFile)

DataProvider creates the import archive

creates one task for each target language out of the one task (project) previously saved

creates and calls **editor_Models_Import::import**(DataProvider)

uses the DataProvider to get and check the import data

triggers event *afterUploadPreparation*

sets several data in task and in **editor_Models_Import_Configuration**

locks the task with state import

triggers event *beforeImport*

prepares the import workers

editor_Models_Import_Worker_FileTree
editor_Models_Import_Worker_ReferenceFileTree
editor_Models_Import_Worker

triggers event *importWorkerQueued*

prepares the import end workers

editor_Models_Import_Worker_SetTaskToOpen sets the task to its initial state (open|unconfirmed), runs only on a successful import

editor_Models_Import_Worker_FinalStep (runs always, also in the case of import errors)

triggers event *importCompleted* notifies the GUI that the import is done (regardless of state error)

run the workers

called workers in detail:

editor_Models_Import_Worker_FileTree

imports meta data from the import directory via **editor_Models_Import_MetaData** default meta data is listed below, new meta data importers can be added by plugins.

editor_Models_Import_MetaData_QmSubsegments import XML MQM Subsegment definitions

editor_Models_Import_MetaData_PixelMapping import PixelMapping width definitions

editor_Models_Import_TermListParser_Tbx import TBX data from the import package for termtagging and associate a new term collection to the task (if TBX provided in ZIP)

trigger event *beforeDirectoryParsing*

editor_Models_Import_FileList reads the available files and creates the working files (formerly review files), relais files and reference files file trees, by using

editor_Models_Import_DirectoryParser_WorkingFiles

trigger event *beforeFileNodeCreate*

editor_Models_Foldertree

editor_Models_Import_DirectoryParser_ReferenceFiles

editor_Models_RelaisFoldertree

editor_Models_Foldertree_SyncToFiles

trigger event *afterDirectoryParsing*

editor_Models_Import_Worker_ReferenceFileTree

editor_Models_Import_FileList

editor_Models_Import_DirectoryParser_ReferenceFiles **possible bug, since called also in editor_Models_Import_Worker_FileTree T homas Lauria**

editor_Models_Import_Worker

checks the serialized parameters

calls **editor_Models_Import_Worker_Import::import**

trigger event *beforeImportFiles*

import work files (**importFiles**)

loads the found files via **editor_Models_Foldertree**

loops over each to be imported file and

uses **editor_Models_File_FilterManager** to apply filters on each file

parses each file with a suitable FileParser:

editor_Plugins_VisualReview_XmlXsl_AlignParser

editor_Models_Import_FileParser_Transit

editor_Models_Import_FileParser_Xml

editor_Models_Import_FileParser_Xlf

editor_Models_Import_FileParser_Sdlxliff

editor_Models_Import_FileParser_Csv

editor_Models_Import_FileParser_XlfZend

editor_Models_Import_FileParser_DisplayTextXml

uses the **segment processors** to save each found segment

editor_Models_Import_SegmentProcessor_MqmParser

editor_Models_Import_SegmentProcessor_RepetitionHash

editor_Models_Import_SegmentProcessor_Review

syncFileOrder syncs the order number of the files to the newly created segments

import relais files (**importRelaisFiles**)

same logic as for importFiles, just using the relais files and **editor_Models_Import_SegmentProcessor_Relais** instead editor_Models_Import_SegmentProcessor_Review

creates the **materialized view** for the task

calculateMetrics (word and segment count)

editor_Workflow_Manager::initDefaultUserPrefs

trigger event *importCleanup*

Plugins

VisualReview

- invokes on **editor_Models_Import[*afterUploadPreparation*]:** prepares HTML via URL based visual review
- invokes on **editor_Models_Import_Worker_FileTree[*beforeDirectoryParsing*]:** prepares visual source files
- invokes on **editor_Models_Import_Worker_FileTree[*afterDirectoryParsing*]:** prepares and checks optional meta data for visual review (XLST etc).
- invokes on **editor_Models_Import[*afterImport*]:** queues the workers doing the visual review parts of the import:
    - **editor_Plugins_VisualReview_PdfToHtmlWorker**
    - **editor_Plugins_VisualReview_WgetHtmlWorker**
    - **editor_Plugins_VisualReview_HtmlImportWorker**
    - **editor_Plugins_VisualReview_XmlXsltToHtmlWorker**
    - **editor_Plugins_VisualReview_SegmentationWorker**

TermTagger

- invokes on **editor_Models_Import[*afterImport*]:** starts the termtagging by queuing the **editor_Plugins_TermTagger_Worker_TermTaggerImport** workers
- invokes on **editor_Models_Import_SegmentProcessor_Review[*process*]:** sets each non editable segment to be ignored from termtagging (contrable via config)
- invokes on **editor_Models_Import_MetaData[*importMetaData*]:** adds the **editor_Models_Import_TermListParser_Tbx** importer to the MetaData importer list
- invokes on **editor_TaskController[*afterPostAction*]:** prepares HTML via URL based visual review

Okapi

- invokes on **editor_Models_Import_Worker_FileTree[*beforeDirectoryParsing*]:** reconfigures the import configuration based on the existence of a custom bconf file
- invokes on **editor_Models_Import_Worker_FileTree[*afterDirectoryParsing*]:** queues a **editor_Plugins_Okapi_Worker** worker for each file which is not processable with a local fileparser or filefilter. The worker sends the file to Okapi and waits until Okapi has processed the file.
- invokes on **editor_Models_RelaisFoldertree[*customHandleFile*]:** prepares the relais file filenames so that they match to the filenames with the added xlf suffix.
- invokes on **editor_Models_Import_Worker_Import[*importCleanup*]:** archives the files coming from Okapi by creating the OkapiArchive.zip

MatchAnalysis

- invokes on **editor_TaskController[*analysisOperation*]:** queues **editor_Plugins_MatchAnalysis_BatchWorker** if enabled, and **editor_Plugins_MatchAnalysis_Worker** which is doing the analysis
- invokes on **editor_TaskController[*pretranslationOperation*]:** queues **editor_Plugins_MatchAnalysis_BatchWorker** if enabled, and **editor_Plugins_MatchAnalysis_Worker** which is doing the analysis with pre-translation

LockSegmentsBasedOnConfig

- invokes on **editor_Models_Import[*afterImport*]:** does segment locking based on a flexible config