Worker architecture: load balancing, asynchronous processing, queuing and chaining of jobs

translate5 has a core component for this. It is able to manage multiple processes at the same time.

This component e. g. is used for

- importing and exporting files
- marking terminology
- counting terminology errors, words and characters
- etc.

Worker capabilities

Asynchronous processing

The GUI is not blocked by the server side processes. The user gets informed, when the background processes finish

Chaining of workers

This refers to different worker types, that should run over the data of the same task.

The developer can define, in which order background processes should run over the the segments. E. g. first pre-translation, then termTagging, then segment analysis. In this case the first termTagging worker for a task will only start, after the last pre-translation worker is finished.

Queuing of workers

This refers to the same worker type running in parallel on multiple tasks.

It can be defined, how many parallel processes of which worker type are allowed. E. g. max. 5 parallel import processes (because the machine needs free CPU capacity to answer user GUI requests) but additional 50 parallel MT-pretranslation requests (because MT-pretranslation runs on a different server). If there are more parallel requests than allowed, the additional ones get queued and will be started, as soon as there is free capacity.

Load balancing

This refers the same worker type running in parallel multiple times on the same task.

translate5 can e. g. talk to 5 different instances of an external system (number configurable) at the same time and pass chunks of the job to each of them (the size of the chunks is configurable, e. g. 50 segments per chunk).

When one chunk is returned by one of the external instances, the next chunk will be passed to it. This continues until all segments have been passed to the external system.

Worker architecture - developer notes

Worker can be external processes called via REST, or internal translate5 processes.

The basic classes you can build on to use these mechanisms are

- ZfExtended_Worker_Abstract
- ZfExtended_Models_Worker

Have a look at the termTagger-Plugin (located in /application/editor/Plugins/TermTagger) as a complex example and the LockSegmentsBasedOnConfig-Plugin as a simple example (located in /application/editor/Plugins/LockSegmentsBasedOnConfig).

The basic idea is, that each one process mostly runs completely independent (kind of an own thread) from the others. Such a process is called a "worker" in translate5. The workers can be called directly by a php-class (then it does not run as as an own process, but as part of the calling php-class) or they can be queued in the DB-table Zf_worker.

PauseImportWorker

The worker was designed to pause another worker until certain conditions are met.

If you need to wait until something is happened or finished before starting a particular worker do next:

- Create a new worker class extending MittagQI\Translate5\Import\PauseImportWorker and make it to be a dependency of the worker that you want to pause

- Create a class implementing PauseWorkerProcessorInterface

- Add your just created PauseWorker to be queued with a worker that you want to pause

For more information about the implementation take a look PauseMatchAnalysisWorker

The queue and load balancing

A worker is queued by calling the ZfExtended_Worker_Abstract-Method "queue". This method also kicks of the worker queue, this means starts the first worker in the queue. Depending on what is defined for a child class of ZfExtended_Worker_Abstract, the queue calls one or multiple resources for this kind of worker. For example the editor_Plugins_TermTagger_Worker_Abstract distributes the termTagging-Load to as many different TermTagger-REST-servers, as are defined in Zf_configuration for the termTagger.

Keep in mind that translate5 worker system can start, in some cases, more workers than it actually need to processneed the certain number of segments. For example, we have:

1. Config option **`runtimeOptions.plugins.SpellCheck.languagetool.url.import`** is set as **["http://localhost:8081/v2", "http://localhost:8082/v2"]** e.g two slots (LanguageTool-REST servers) available
2. Config option **`runtimeOptions.worker.editor_Plugins_SpellCheck_Worker_Import.maxParallelWorkers`** is set as **2**
3. `editor_Plugins_SpellCheck_Configuration::IMPORT_SEGMENTS_PER_CALL = 5`

4. Some task containing 10 segments

The above input conditions mean that system need 2 workers to process the task, but in fact 3 workers will be queued, because:

1. first one will be queued in editor_Plugins_SpellCheck_QualityProvideraddWorker() by calling $workerqueue() with *no args*
2. other two - in editor_Models_Import_Worker_ResourceAbstract->queue($parentId = 0, $state = null, *$startNext = true*), e.g. one worker per each slot defined

So there can be the case when no real parallelization is achieved due to that first worker can be set up with slot 1, second with slot 1 and third with slot 2, But as long as it's happening randomly, this is not always the case.

But the much more important good news here is that such a lack of parallelization is in fact only affects small tasks, because the maximum negative effect is that the **only first chunk** of segments can be processed with no real parallelization. So the above notes are just for developers who are trying to figure out how to explain what's going on in the background, because developers may use small tasks for debugging and this thing may produce questions.

Tip: A worker can queue another worker of the same type (or another type) before exiting. This way a worker chain can be established, working itself through complex and long tasks.

The queue and dependencies

Each worker can have dependencies, defined in the DB-table Zf_worker_dependencies. If a dependency is defined, this worker will not be started as long as there are dependent workers belonging to the same task in the queue. This way different kinds of workers can build on the results of each other.

If you plan to develop a translate5 plugin based on workers, please contact the translate5 core development team for guidance. Workers are complex and need to be implemented in the right way.