TMEngine

An Open Source Translation Memory Manager

Table of Contents

Overview

TMEngine

TMEngine is an open source Translation Memory (TM) manager written in Java.

TMEngine can be used in two ways:

- As an embedded library that manages translation memories in a Java application;
- As a standalone TM server via its REST API.

Running as a Standalone Server

Starting the Server

Running `.\tmserver.bat` or `./tmserver.sh` without parameters displays help for starting TMEngine as a standalone server.

```
Usage:

                tmserver.sh [-help] [-version] [-port portNumber]

                Where:

                -help:      (optional) Display this help information and exit
                -version:   (optional) Display version & build information and exit
                -port:      (optional) Port for running HTTP server. Default is 8000
```

You can verify that the server is running by visiting its default web page: http://localhost:8000/TMServer/ (adjust port number if you change it).

REST API

The REST methods that TMEngine's server supports are:

- Create Memory
- List Memories
- Open Memory
- Close Memory
- Import TMX File
- Process Status
- Export TMX File
- Search Translations
- Concordance Search
- Rename Memory
- Delete Memory
- Stop Server

Default TMEngine URL is '`http://localhost:8000/TMServer/`'.

Note

It is possible to select a custom port for the server, passing the '`-port`' parameter to the script used for launching it.

All methods return a JSON object with a '`status`' field. Applications must watch this field and verify that it is set to '`OK`'.

In case of error, the JSON response includes a field named '`reason`' that contains the error cause.

Create Memory

End Point: `[TMEngine URL]/create`

Default: `http://localhost:8000/TMServer/create`

Send a '`POST`' request to the method end point with these parameters in a JSON body:

Field	Required	Content
`id`	No	ID of the memory to create. The value of '`id`' must be unique. Default value is current server time represented as the number of milliseconds since January 1, 1970, 00:00:00 GMT
`name`	Yes	A meaningful name to identify the memory
`owner`	No	Text string used to identify the owner of the memory. Default value is the login name of the user running the server.
`type`	No	Type of engine to use. Possible values are: • '`MapDbEngine`' (default) • '`SQLEngine`'
`serverName`	No	Name or IP of the server running MySQL or MariaDB. Required for `SQLEngine`. Defaut value: '`localhost`'
`port`	No	Port in which MySQL or MariaDB listens for requests. Required for `SQLEngine`. Default value: `3306`
`userName`	No	ID of of the MySQL or MariaDB user creating the database. Required for `SQLEngine`.
`password`	No	Password of the MySQL or MariaDB user creating the database. Required for `SQLEngine`.

Example:

```
{
  "name": "First Memory",
  "type": "MapDbEngine"
}
```

```
{
  "name": "MariaMemory",
  "type": "SQLEngine",
  "serverName": "localhost",
  "port": 3306,
  "userName": "root",
```

```
  "password": "secret123!"
}
```

The server responds with a JSON object containing two fields.

On success, field '`status`' is set to '`OK`' and field '`id`' contains the ID assigned to the new memory.

Example:

```
{
  "status": "OK",
  "id": "1234567890987"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Duplicated id"
}
```

List Memories

End Point: `[TMEngine URL]/list`

Default: `http://localhost:8000/TMServer/list`

Send a '`GET`' request to the method end point.

The server responds with a JSON object containing two fields. On success, field '`status`' is set to '`OK`' and field '`memories`' contains an array with memory details.

```
{
  "memories": [
     {
       "owner": "manager",
       "isOpen": false,
       "name": "Fluenta Localization",
       "id": "fluenta",
       "type": "MapDbEngine",
       "creationDate": "2019-09-10 21:54:13 UYT"
     },
     {
       "owner": "manager",
       "isOpen": false,
       "name": "First Memory",
       "id": "1568163112478",
       "type": "MapDbEngine",
       "creationDate": "2019-09-10 21:51:52 UYT"
     }
  ],
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Error reading memories"
}
```

Open Memory

End Point: `[TMEngine URL]/create`

Default: `http://localhost:8000/TMServer/open`

Send a '`POST`' request to the method end point with this parameter in a JSON body:

Field	Required	Content
`id`	Yes	ID of the memory to open

Example:

```
{
  "id": "1568163112478"
}
```

The server responds with a JSON object. On success, field '`status`' is set to '`OK`'.

```
{
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Unknown memory type"
}
```

Close Memory

End Point: `[TMEngine URL]/create`

Default: `http://localhost:8000/TMServer/close`

Send a '`POST`' request to the method end point with this parameter in a JSON body:

Field	Required	Content
`id`	Yes	ID of the memory to close

Example:

```
{
  "id": "1568163112478"
}
```

The server responds with a JSON object. On success, field '`status`' is set to '`OK`'.

```
{
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Unknown memory"
}
```

Import TMX File

End Point: `[TMEngine URL]/import`

Default: `http://localhost:8000/TMServer/import`

Send a '`POST`' request to the method end point with these parameters in a JSON body:

Field	Required	Content
`id`	Yes	ID of the memory to populate with TMX data
`file`	Yes	Path to the TMX file being imported
`subject`	No	Name or identifier of the subject associated with the TMX file
`client`	No	Name or identifier of the client associated with the TMX file
`project`	No	Name or identifier of the project associated with the TMX file

---

Note

The TMEngine server must have access to the TMX file being imported. When importing a TMX file into a remote server, copy or upload the file to the server first and supply the right path in the JSON body.

Example:

```
{
  "id": "1568163112478",
  "file": "/Volumes/Data/segments.tmx",
  "project": "Main TM"
}
```

The server responds with a JSON object containing two fields.

On success, field '`status`' is set to '`OK`' and field '`process`' contains the ID of the background import process that was initiated.

```
{
  "process": "1568222345643",
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

```
{
  "status": "failed",
  "reason": "Unknown memory type"
}
```

After starting the import process, monitor its status using the Process Status method. On successful completion, the result will contain the number of segments imported.

Example:

```
{
  "result": "Completed",
  "data": {
    "imported": "57678"
  },
  "status": "OK"
}
```

Process Status

End Point: `[TMEngine URL]/status`

Default: `http://localhost:8000/TMServer/status`

Send a `POST` request to the method end point with this parameter in a JSON body:

Field	Required	Content
`process`	Yes	ID of the background process to check

Example:

```
{
  "process": "1568223016762"
}
```

The server responds with a JSON object.

On successful status check, field '`status`' is set to '`OK`' and field '`result`' contains current status.

Example:

Field '`result`' may have these values:

- **Pending**: processing is still going on.

  ```
  {
    "result": "Pending",
    "status": "OK"
  }
  ```

- **Completed**: processing has finished. If the process produces any data, it is placed in the '`data`' field.

```
{
  "result": "Completed",
  "data": {
    "imported": "57678"
  },
  "status": "OK"
}
```

- **Failed**: processing failed. Failure reason is provided in '`reason`' field.

```
{
  "result": "Failed",
  "reason": "/Volumes/Data/something.tmx (No such file or directory)",
  "status": "failed"
}
```

If process status cannot be checked, the server omits the '`result`' field and provides a failure reason.

```
{
  "reason": "Missing 'process' parameter",
  "status": "failed"
}
```

Export TMX File

End Point: `[TMEngine URL]/create`

Default: `http://localhost:8000/TMServer/export`

Send a '`POST`' request to the method end point with these parameters in a JSON body:

Field	Required	Content
`id`	Yes	ID of the memory to populate with TMX data
`file`	Yes	Path to the TMX file being created
`langs`	No	JSON array contaning the list of languages to export
`srcLang`	No	Language to set as source langage. The wildcard '`*all*`' is used by default
`properties`	No	JSON object with string properties to set in the exported file

> **Note**
>
> when exporting a TMX file on a remote server, make sure the TMEngine server has access to the specified location.

Example:

```
{
  "id": "1568163112478",
  "file": "/Volumes/Data/segments.tmx",
  "langs": [
      "en-US",
       "ja",
       "fr-FR",
       "it"
  ],
  "srcLang": "en-US",
  "properties": {
    "project": "Milky Way",
    "subject": "Astronomy Device"
  }
}
```

The server responds with a JSON object containing two fields.

On success, field '`status`' is set to '`OK`' and field '`process`' contains the ID of the background export process that was initiated.

```
{
  "process": "1568222345643",
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

```
{
  "status": "failed",
  "reason": "Unknown memory type"
}
```

After starting the export process, monitor its status using the Process Status method.

Search Translations

End Point: `[TMEngine URL]/create`

Default: `http://localhost:8000/TMServer/search`

Send a '`POST`' request to the method end point with these parameters in a JSON body:

Field	Required	Content
`id`	yes	ID of the memory where the search should be performed
`text`	Yes	Text string to search
`srcLang`	Yes	Source language code
`tgtLang`	Yes	Target language code
`similarity`	Yes	Integer value indicating the lowest similarity percentage to include in results
`caseSensitive`	Yes	Boolean value indicating whether the search should be case sensitive or not

Example:

```
{
  "id": "1572538708492",
  "text": "tax compliance",
  "srcLang": "en-GB",
  "tgtLang": "fr-FR",
  "similarity": 70,
  "caseSensitive": false
}
```

The server responds with a JSON object containing two fields.

On success, field '`status`' is set to '`OK`' and field '`process`' contains the ID of the background import process that was initiated.

```
{
  "process": "1572531573026",
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

```
{
  "status": "failed",
  "reason": "Unknown memory type"
}
```

After starting the import process, monitor its status using the Process Status method.

On successful completion, the result will contain an array of similar segments in the `data` field.

Example:

```
{
  "result": "Completed",
  "data": {
    "matches": [
      {
        "similarity": 71,
        "origin": "1572538708492",
        "source": "<tuv xml:lang="en-GB"><seg>Non-compliance</seg></tuv>",
        "target": "<tuv xml:lang="fr-FR"><seg>Violation</seg></tuv>",
        "properties": {
          "creationdate": "20070126T082848Z",
          "subject": "Taxes",
          "x-Origin": "TM",
          "project": "Main TM",
          "changedate": "20070126T082848Z",
          "tuid": "1546700322331",
          "creationid": "MC",
          "changeid": "MC",
          "lastusagedate": "20070126T082848Z",
          "customer": "ACME Auditors"
        }
```

```
      },
      {
        "similarity": 73,
        "origin": "1572538708492",
        "source": "<tuv xml:lang="en-GB"><seg>Legal Compliance</seg></tuv>",
        "target": "<tuv xml:lang="fr-FR"><seg>Conformité légale</seg></tuv>",
        "properties": {
          "creationdate": "20160725T141611Z",
          "x-ConfirmationLevel": "ApprovedTranslation",
          "subject": "Taxes",
          "x-Origin": "TM",
          "project": "Main TM",
          "changedate": "20160727T093143Z",
          "tuid": "1546700366038",
          "creationid": "Aqcis9\Aqcis",
          "changeid": "FG",
          "lastusagedate": "20160727T093143Z",
          "customer": "ACME Auditors"
        }
      },
      {
        "similarity": 100,
        "origin": "fluenta",
        "source": "<tuv xml:lang="en-GB"><seg>tax compliance</seg></tuv>",
        "target": "<tuv xml:lang="fr-FR"><seg>Conformité fiscale</seg></tuv>",
        "properties": {
          "creationdate": "20171004T111450Z",
          "subject": "Taxes",
          "project": "Main TM",
          "changedate": "20171004T111450Z",
          "tuid": "1546700370945",
          "changeid": "translator2",
          "usagecount": "1",
          "x-ConfirmationLevel": "Translated",
          "x-Origin": "TM",
          "creationid": "translator2",
          "lastusagedate": "20171006T103930Z",
          "customer": "ACME Auditors"
        }
      }
    ],
  },
  "status": "OK"
}
```


Concordance Search

End Point: `[TMEngine URL]/concordance`

Default: `http://localhost:8000/TMServer/concordance`

Send a '`POST`' request to the method end point with these parameters in a JSON body:

Field	Required	Content
`id`	yes	ID of the memory where the search should be performed
`text`	Yes	Text string to search
`srcLang`	Yes	Source language code
`limit`	Yes	Integer value indicating the maximum number of matches to include
`isRegexp`	Yes	Boolean value indicationg wether the search text should be treated as a regular expression
`caseSensitive`	Yes	Boolean value indicating whether the search should be case sensitive or not

Example:

```
{
  "id": "fluenta",
  "text": "segment",
  "srcLang": "en",
  "limit": 5,
  "isRegexp": false,
  "caseSensitive": true
}
```

On success, field '`status`' is set to '`OK`' and field '`process`' contains the ID of the background import process that was initiated.

```
{
  "process": "1572531573026",
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

```
{
  "status": "failed",
  "reason": "Unknown memory type"
}
```

After starting the import process, monitor its status using the Process Status method.

On successful completion, the result will contain an array of <tu> elements that contain the searched text in the `data` field.

Example:

```
{
  "result": "Completed",
  "data": {
    "entries": [
      "<tu creationid="rmraya" creationdate="20161225T150949Z" creationtool="Swordfish"
      creationtoolversion="3.3-8" tuid="-1247472893-0-1586928971">
      <prop type="project">Fluenta</prop>
      <tuv xml:lang="es"><seg>Hay segmentos con errores de etiquetas.</seg></tuv>
```

```
      <tuv xml:lang="en"><seg>There are segments with tag errors.</seg></tuv></tu>"
    ],
  },
  "status": "OK"
}
```

Rename Memory

End Point: `[TMEngine URL]/rename`

Default: `http://localhost:8000/TMServer/rename`

Send a '`POST`' request to the method end point with these parameters in a JSON body:

Field	Required	Content
`id`	Yes	ID of the memory to rename
`name`	Yes	New name for the memory

> **Note**
>
> Only memories of type `MapDbEngine` can be renamed.

Example:

```
{
  "id": "1568163112478",
  "name": "Updated Memory Name"
}
```

The server responds with a JSON object containing two fields.

On success, field '`status`' is set to '`OK`'.

Example:

```
{
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Wrong memory type"
}
```

Delete Memory

End Point: `[TMEngine URL]/delete`

Default: `http://localhost:8000/TMServer/delete`

Send a '`POST`' request to the method end point with this parameter in a JSON body:

Field	Required	Content
`id`	Yes	ID of the memory to delete

Example:

```
{
  "id": "1568163112478"
}
```

The server responds with a JSON object. On success, field '`status`' is set to '`OK`'.

```
{
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Unknown memory"
}
```

Stop Server

End Point: `[TMEngine URL]/create`

Default: `http://localhost:8000/TMServer/stop`

Send a '`GET`' request to the method end point.

The server responds with a JSON object. On success, field '`status`' is set to '`OK`'.

Example:

```
{
  "status": "OK"
}
```

On error, field '`status`' is set to '`failed`' and field '`reason`' contains the error cause.

Example:

```
{
  "status": "failed",
  "reason": "Error connecting to database"
}
```

Java Library