

**FINAL PROJECT: Translation API**

**BY**

**Prachi Chandanshive**

**Student ID: 801203237**

## **Index:**

1. About the project:	..... 3
2. Vendor api	..... 4
3. Api structure	..... 7
4. Project Urls:	..... 9
5. Output	..... 10

## **1. About the project:**

Translation API project focuses on integrating the azure cognitive services translator with the help of various REST APIs. The REST APIs used in this project are used to utilize various services provided by the azure translator.

With systematic documentation using swagger in this project, one can easily understand all the azure translate services offered by azure translate API.

### **Regarding the AZURE TRANSLATOR API:**

Translator API by azure is flexible and can be used with many operating systems. The features available for this api include:

- Text translation:  
Translate a text from one source language to another targeted language.
- Document Translation:  
Translates list of documents from one language to targeted language protecting the formatting of the document.

URL For Azure API:

<https://azure.microsoft.com/en-us/services/cognitive-services/translator/>

### **Endpoints:**

Two endpoints are generated for the translator api:

- a. Text endpoint:

The text endpoint can be used along with the API key from the node js project to translate text, detect text language and transliterate texts into scripts.

- b. Document endpoint:

The custom document endpoint created with the resource in azure can be used with API KEY to translate documents.

For getting endpoints, creating a translation service is required from a Microsoft Account.

This will provide two endpoints for translation of text and other for documenty.

## 2. Implementation of Project:

The project is implemented using ExpressJS and the api documentation is implemented using swagger.

ExpressJS is widely used for developing web applications easily. It provides various api methods like get, put, delete which can be used while developing APIs along with implementation of routing is easier with the built-in packages provided. Performance of application using ExpressJs is powerful as it is scalable to large volume of data and can perform operations asynchronously.

The project takes care of data sanitization and validation with the help of express-validator package. This package provides various methods to check data ranging from request body to parameters and thus helps in maintaining the securing of the application.

In the project, different APIs are created which calls to different azure translation service API, provides the input parameters required to that API and gets the expected output.

To make the code and application secure, the API\_KEY for the project is stored in the process environment variables of the application which can be only given through cmd line while running the project and not exposed directly in the code.

The list of APIs and their functionality:

Sr. No	API urls	functionality
1.	/translatetxtByCode	Returns text translated in the given language code
2.	/documentTranslate	Returns the translated documents response and stores documents in the target container specified.
3.	/translatefrmEng	Returns text from English translated in the given language code
4.	/detectAndTranslate	Returns text translated in the specified code parameter for language after automatically detecting the language of the text



5.	/detectLanguage	Returns language detected, confidence score of the language detected (closer to 1.0 means strongly confident), is translation and transliteration supported as well.
6.	/onlyTransliterate	Returns the script and output text.
7.	/sentenceLength	Returns the character count from the text given in request body.
8.	/searchdictionary	Returns the list of various translations including confidence score, optimized text for display, part of speech(post tag) and information about previous translation .
9.	/getExamples	Returns the list of translations including confidence score, optimized text for display, part of speech(post tag) and information about previous translation .
10.	/getDocumentFormats	Returns the the status of a specific document in a Document Translation request .
11.	/getJobStatus	Returns the the status of a specific document in a Document Translation request .

For document translation:




Two blob containers are created using the Microsoft account.


1. Source container is called inputdocs  
The permission for this container is read and list.
2. Destination container is called translateddocs  
The permissions for this container is write and list.

Home > analyticsforstorage

analyticsforstorage | Containers  ... 

Storage account

Search (Cmd+/) « + Container  Change access level  Restore containers v Refresh  Delete

Search containers by prefix  Show deleted containers

Name	Last modified	Public access level	Lease state	
<input type="checkbox"/> \$logs	4/30/2022, 11:23:38 PM	Private	Available	...
<input checked="" type="checkbox"/> inputdocs	4/30/2022, 11:29:59 PM	Private	Available	...
<input checked="" type="checkbox"/> translateddocs	4/30/2022, 11:30:28 PM	Private	Available	...

Overview

Activity log

Tags

Diagnose and solve problems

Access Control (IAM)

Data migration

Events

Storage browser (preview)

Data storage

Containers

File shares

Queues

Tables

Fig: inputdocs and translateddocs container

### 3. API Structure:

The input parameters of the API mostly consist of:

POST /translateTextByCode

Returns text translated in the given language code

Parameters Try it out

Name	Description
<b>body</b> <small>required</small> array (body)	text that needs to be translated,code of language in which the text needs to be translate and language code of the text that needs to be translated  Example Value   Model <pre>{  "txt": "string",  "code": [    "string"  ],  "fromLang": "string"}</pre> Parameter content type application/json

text – text that needs to be translated

code- array of language codes in which it can be translated.

fromLang – language code in which the text is

```
{  "txt": "string",  "code": [    "string"  ],  "fromLang": "string"}
```

The response for the api :

Responses		Response content type
		application/json
Code	Description	
200	Successfully response returned	
500	Internal Error	

Two status code are used : 200 if the response is successfully returned else 500 response will be returned along with its respective error message.

This project provides easy to understand and test API documentation for utilizing translate service of azure hence saving the efforts and time spend in understanding the service.

default ^	
POST	/translatetxtByCode
POST	/documentTranslate
POST	/translatefrmEng
POST	/detectAndTranslate
POST	/detectLanguage
POST	/onlyTransliterate
POST	/sentenceLength
POST	/searchdictionary
POST	/getExamples
GET	/getDocumentFormats
GET	/getJobStatus
Models	



#### 4. Project urls:

<b>Sr No.</b>	<b>URL</b>	<b>Function</b>
1	<a href="http://143.198.115.87:3000/">http://143.198.115.87:3000/</a>	Project base url
2	<a href="http://143.198.115.87:3000/docs">http://143.198.115.87:3000/docs</a>	Swagger Api documentation
3	<a href="https://github.com/prachic1210/SIFinalProject">https://github.com/prachic1210/SIFinalProject</a>	GitHub code repository

## 5. Outputs:

### a. Testing api '/translatetxtByCode'

#### Using swagger:

**POST** /translatetxtByCode

Returns text translated in the given language code array

**Parameters** Cancel

Name	Description
<b>body</b> <span>required</span> array (body)	text that needs to be translated,code of language in which the text needs to be translate and language code of the text that needs to be translated <div>Edit Value   Model</div> <pre>{   "txt": "welcome here",   "code": {     "fr","ko"   },   "fromLang": "en" }</pre>

Cancel

Parameter content type  
application/json

#### Response:

Request URL  
http://143.198.115.87:3000/translatetxtByCode

Server response

Code	Details
200	<div><b>Response body</b></div> <div>Translated successfully text by language code[ {   "translations": [     {       "text": "bienvenue ici",       "to": "fr"     },     {       "text": "여기에 오신 것을 환영합니다",       "to": "ko"     }   ] } ]</div> <div><span>Download</span></div>

**Response headers**access-control-allow-origin: \*  
connection: keep-alive  
content-length: 315  
content-type: text/html; charset=utf-8  
date: Mon, 02 May 2022 01:02:50 GMT  
etag: W/"11b-Mb4SDBLMtMqRePs+f8aA2lEPoo"  
keep-alive: timeout=5  
x-powered-by: Express

## Using postman:

The screenshot shows a Postman interface for a POST request to `http://143.198.115.87:3000/translateTxtByCode`. The request body is set to `x-www-form-urlencoded` and contains the following data:

Key	Value
txt	cloud
code	ko
fromLang	en
code	fr
toLang	es
translation	tiburón

The response status is `200 OK` with a time of `249 ms` and a size of `536 B`. The response body is displayed in the `Preview` tab:

```
Translated successfully text by language code[ { "translations": [ { "text": "구름", "to": "ko" }, { "text": "nuage", "to": "fr" } ] } ]
```

b. Document translated using api /documentTranslate:

## Using swagger,

Initially inputdocs container have the following two documents both in english language:

The screenshot shows the Azure portal interface for the `inputdocs` container. The container contains two documents:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
interlude.docx	5/1/2022, 5:38:34 PM	Hot (Inferred)		Block blob	11.99 KiB	Available
intro.docx	4/30/2022, 11:30:57 ...	Hot (Inferred)		Block blob	11.63 KiB	Available

After executing the API,

Response generated:

The screenshot displays a REST client interface with the following details:

- Request URL:** `http://143.198.115.87:3000/documentTranslate`
- Server response:**
  - Code:** 200
  - Response body:** `Documents translated successfully ""`
  - Response headers:**

```
access-control-allow-origin: *
connection: keep-alive
content-length: 36
content-type: text/html; charset=utf-8
date: Mon, 02 May 2022 01:09:11 GMT
etag: W/"24-U/SuRWGerssiga+IzPWfPubDA"
keep-alive: timeout=5
x-powered-by: Express
```

Buttons for 'Download' and 'Responses' are visible at the bottom right of the response section.

And translated documents will be stored in translateddocs container:

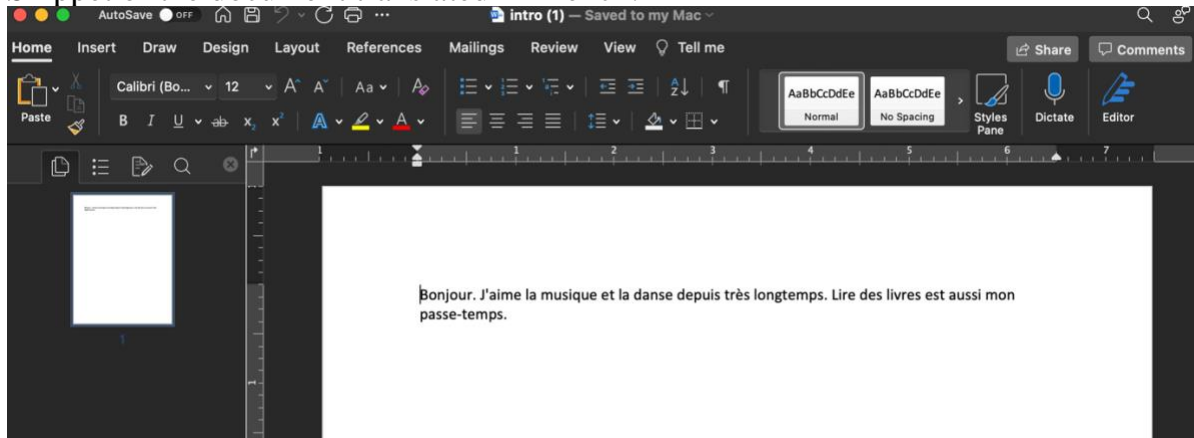
The screenshot shows the Azure portal interface for the 'translateddocs' container. The left sidebar includes navigation links for Overview, Diagnose and solve problems, Access Control (IAM), Settings, Shared access tokens, Access policy, Properties, and Metadata.

The main content area displays the container's overview, including the authentication method (Access key) and location (translateddocs). A search bar for blobs is present, along with a toggle for 'Show deleted blobs'.

A table lists the blobs stored in the container:

Name	Modified	Access tier	Archive status	Blob type	Size	Lease state
<input type="checkbox"/> interlude.docx	5/1/2022, 9:09:20 PM	Hot (Inferred)		Block blob	11.74 KiB	Available
<input type="checkbox"/> intro.docx	5/1/2022, 9:09:20 PM	Hot (Inferred)		Block blob	11.4 KiB	Available

## Snippet of the document translated in French:



## Using postman,

