API-Powered Dictionary Websites

(Software Demonstration)

Sandro Cirulli
Oxford University Press
e-mail: sandro.cirulli@oup.com

Abstract

Oxford University Press (OUP) recently started the Oxford Global Languages (OGL) initiative whose focus is to provide language resources for digitally under-represented languages. In August 2015 OUP launched language websites for isiZulu and Northern Sotho and in March 2016 added websites for Malay and Urdu. The backend of these websites is based on an API retrieving data originally modelled in RDF and delivering data to the frontend in JSON.

The software presentation focuses on the API (Application Programming Interface) developed to power these websites. We show API calls to search dictionary entries, add new content on the website in real-time and delete it if need be. We discuss the advantages of API-powered websites, how the API allowed OUP to crowdsource linguistic data from online communities, and how APIs facilitate the integration of data with external systems and developers. Finally, we outline future work for the next phase of development of the API and OGL websites.

Keywords: API; dictionary; isiZulu; Northern Sotho; Malay; Urdu

1 Introduction

At the end of 2014 Oxford University Press (OUP) launched the Oxford Global Language (OGL) initiative (Oxford University Press, 2016) whose focus is to create linguistic resources particularly for digitally under-represented languages. The aim of the programme is to help language communities over the world create, maintain, and use digital language resources while developing digital-ready content formats to support the growing language needs of technology companies worldwide. The model attempts to create a win-win situation where communities of digitally under-represented languages contribute content, licensees consume data in the digital format they need, and Oxford University Press generates revenue in order to publish language resources online and keep the services free for online communities.

In August 2015 OUP launched its first two language websites for isiZulu[1] and Northern Sotho[2]. In March 2016 OUP added other language websites for Malay[3] and Urdu[4]. The backend of these websites is based on an API (Application Programming Interface) retrieving data originally modelled in RDF (W3C, 2016) and delivering data to the frontend in JSON (ECMA International, 2016).

In the next sections we describe the technical implementation of the backend, discuss the advantages of API-powered websites, and sketch future work for the next development phase of the API and OGL websites.

[1] https://zu.oxforddictionaries.com

[2] https://nso.oxforddictionaries.com

[3] https://ms.oxforddictionaries.com

[4] https://ur.oxforddictionaries.com

2 Technical Implementation

2.1 System Architecture

Figure 1 shows the production system architecture for the OGL language websites.

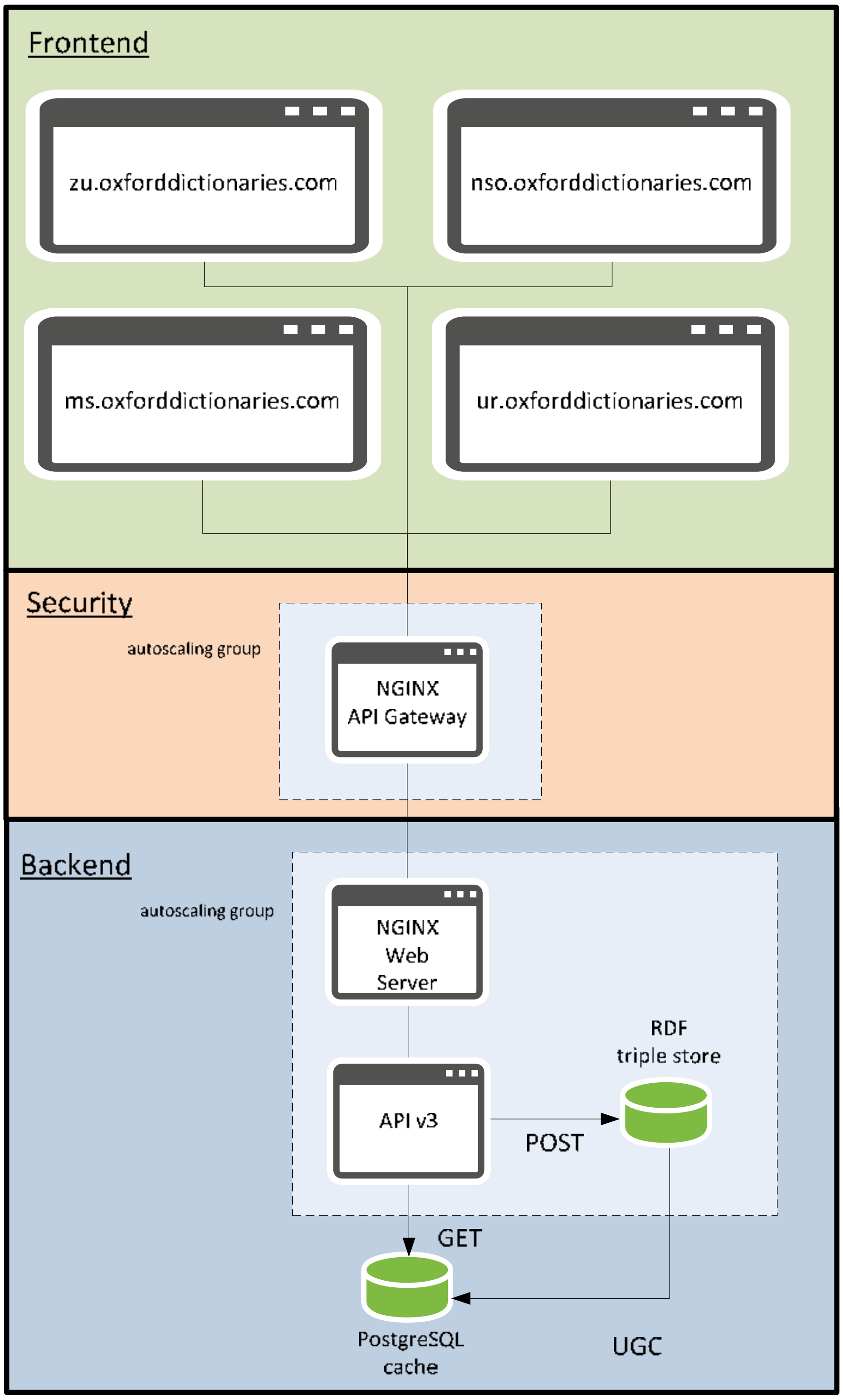


Figure 1: Production System Architecture

The diagram highlights the following layers:

- frontend layer, which relates to the websites frontend (HTML, CSS, JavaScript)
- security layer, which includes authentication and authorization layers handling access and permissions to the API
- backend layer, which relates to the website backend, including the API, the application server, and the data stores

For the frontend and the security layers OUP uses third-party services whereas the backend is fully developed and maintained by OUP and is the focus of the software presentation. In particular, we deployed an NGINX web server for connecting with the API, a RESTful API using Python's Flask web framework for interacting with the data, a PostgreSQL database for caching read-only data, and a GraphDB triple store for storing user generated content (UGC) in RDF and updating the PostgreSQL cache; the whole backend infrastructure runs on Amazon EC2 instances within autoscaling groups (Amazon Web Services, 2016) using a microservices architecture based on Docker (Docker, 2016).

2.2 User Interactions via API

A Web Application Programming Interface (API) is a set of functions, objects, and protocols for exchanging information with a website (Wikipedia, 2016a; Wikipedia, 2016b). An API is essentially a machine-to-machine interface and its typical use is to receive and send data via HTTP requests. For example, a Web API can retrieve data using a HTTP GET request and submit new data via a HTTP POST request.

Figure 2 shows user interactions with the website to add new content (1) and retrieve existing and newly created content (2). In 1.1 the user fills a web form on the frontend and provides information related to an entry (e. g. headword, part of speech, translation, example, etc.). The content of the web form is sent to the API via a POST request (1.2). The API validates the content of the request and generates a SPARQL Update query for the triple store (1.3). The triple store runs the SPARQL Update query (1.4), triggers a process that updates the database cache (1.5), and returns the HTTP status code of the SPARQL query to the API (1.6). The HTTP status code is mapped and transmitted back to the frontend (1.7) which displays a confirmation message to the user (1.8). The new entry created by the user is stored on both the triple store and the database cache and is immediately available on the website.

In 2.1 the user requests an entry via the website search interface. The frontend sends a GET request to the API (2.2). The API translates the request into a SQL query and sends it to the database (2.3). The database runs the SQL query (2.4) and returns the results to the API (2.5). The API serializes the results into a JSON object and returns it to the frontend (2.6). Finally, the website displays the entry to the user in an HTML page (2.7).

In addition to the GET and POST APIs, we developed a DELETE API allowing to remove content via the website's Content Management System (CMS) and a GET API performing a fuzzy match on headwords and inflected forms for auto-completion purposes. The sequence diagram for these API calls is very similar to those illustrated in Figure 2.

During the software demonstration we show these user interactions via our staging website and the API Swagger interface on our developer portal (Figure 3).

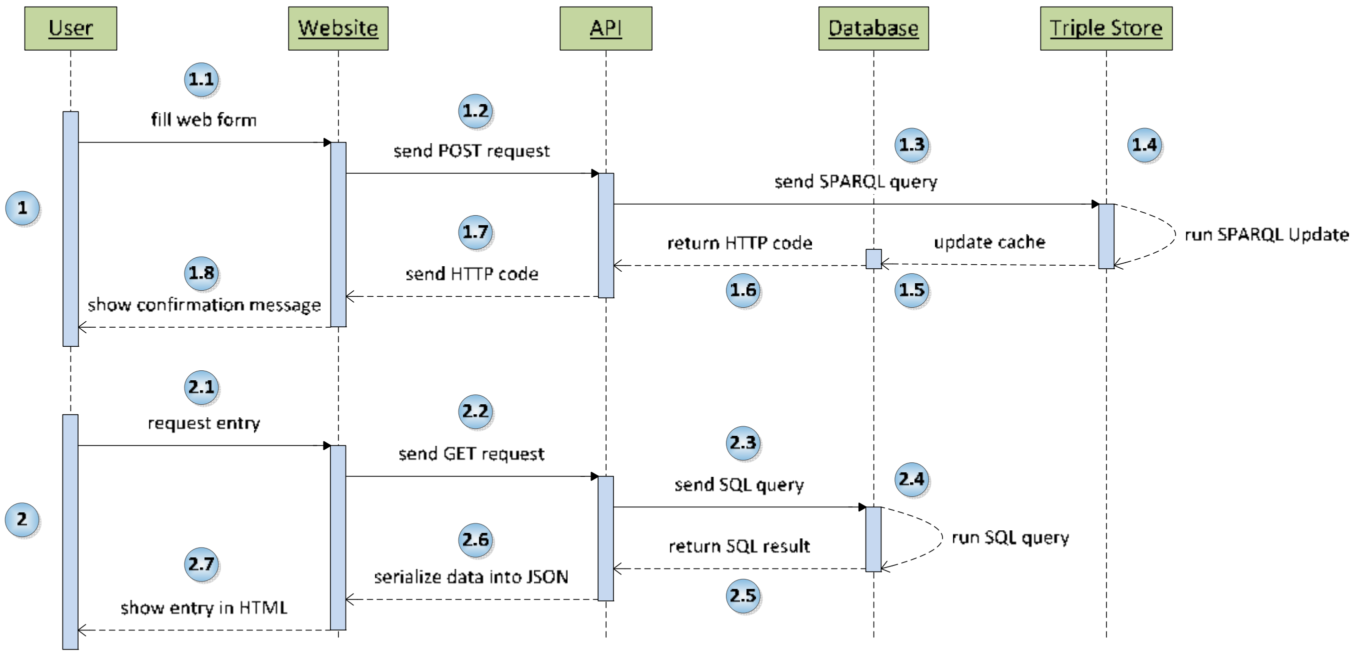
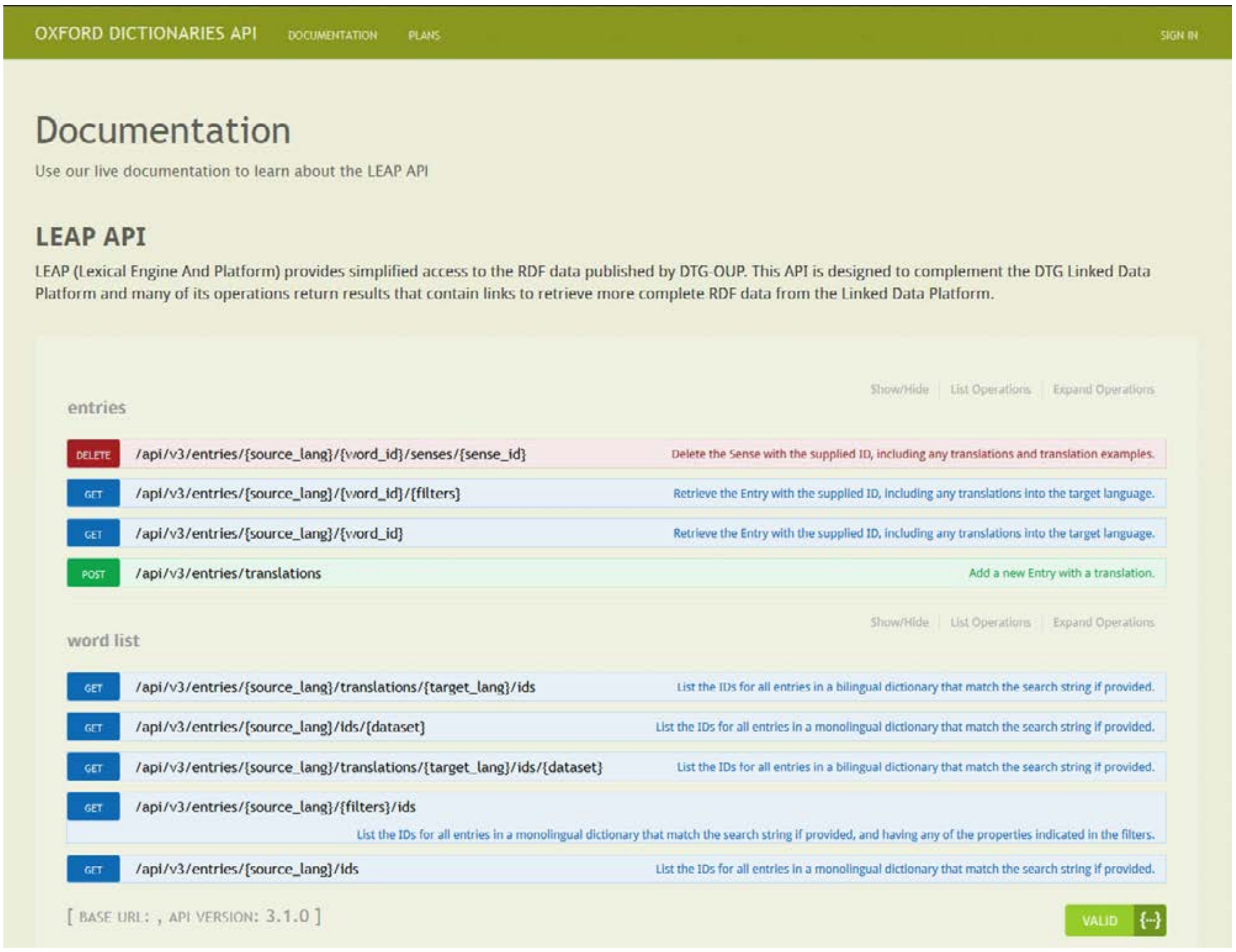


Figure 2: API GET and POST requests



OXFORD DICTIONARIES API DOCUMENTATION PLANS SIGN IN

Documentation

Use our live documentation to learn about the LEAP API

LEAP API

LEAP (Lexical Engine And Platform) provides simplified access to the RDF data published by DTG-OUP. This API is designed to complement the DTG Linked Data Platform and many of its operations return results that contain links to retrieve more complete RDF data from the Linked Data Platform.

entries — Show/Hide | List Operations | Expand Operations

DELETE	/api/v3/entries/{source_lang}/{word_id}/senses/{sense_id}	Delete the Sense with the supplied ID, including any translations and translation examples.
GET	/api/v3/entries/{source_lang}/{word_id}/{filters}	Retrieve the Entry with the supplied ID, including any translations into the target language.
GET	/api/v3/entries/{source_lang}/{word_id}	Retrieve the Entry with the supplied ID, including any translations into the target language.
POST	/api/v3/entries/translations	Add a new Entry with a translation.

word list — Show/Hide | List Operations | Expand Operations

GET	/api/v3/entries/{source_lang}/translations/{target_lang}/ids	List the IDs for all entries in a bilingual dictionary that match the search string if provided.
GET	/api/v3/entries/{source_lang}/ids/{dataset}	List the IDs for all entries in a monolingual dictionary that match the search string if provided.
GET	/api/v3/entries/{source_lang}/translations/{target_lang}/ids/{dataset}	List the IDs for all entries in a bilingual dictionary that match the search string if provided.
GET	/api/v3/entries/{source_lang}/{filters}/ids	List the IDs for all entries in a monolingual dictionary that match the search string if provided, and having any of the properties indicated in the filters.
GET	/api/v3/entries/{source_lang}/ids	List the IDs for all entries in a monolingual dictionary that match the search string if provided.

[BASE URL: , API VERSION: 3.1.0]

VALID {-}

Figure 3: API Swagger Interface

3 Benefits of APIs

The development of an API to power dictionary websites offers the following benefits:

- Reusability: data is accessed via a programmatic method. As a result, additional data for other languages can reuse the same API calls thus reducing development costs in the long term.
- Flexibility: data is delivered in a flexible, modular way and can be shipped in a variety of data formats (XML, JSON, RDF, JSON-LD) through websites, data dumps, web services, etc.
- Crowdsourcing: content contributed by online communities can be easily gathered and shown in real-time on a website. Although this advantage is not unique to APIs, the use of an API facilitates the automation, integration, and reusability of crowdsourced data.
- Integration: external systems, applications, and developers can easily integrate and consume data via APIs.

4 Future Work

In the next years the OGL initiative intends to develop several dictionaries and languages resources especially for digitally under-represented languages. OUP is currently developing the second phase of its programme and plans to launch other dictionary websites built around online communities.
The development of APIs is also a key investment for OUP as it allows to automate processes, clean up data, and speed up content delivery for both web services and licensees. We are also working with commercial and non-commercial partners to open up some datasets via APIs and Semantic Web technologies and would be interested in specific use cases from other potential partners.

5 References

Amazon Web Services (2016). *Amazon EC2 – Virtual Server Hosting*. Accessed at: https://aws.amazon.com/ec2 [29/04/2016].
Docker (2016). *Docker – Build, Ship, Run*. Accessed at: https://www.docker.com [29/04/2016].
ECMA International (2016). *The JSON Data Interchange Format*. Accessed at: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf [29/04/2016].
Oxford University Press (2016). *Oxford Global Languages*. Accessed at: http://www.oxforddictionaries.com/ogl [29/04/2016].
W3C (2016). *Resource Description Framework (RDF)*. Accessed at: https://www.w3.org/RDF [29/04/2016].
Wikipedia (2016a). *Application Programming Interface.* Accessed at: https://en.wikipedia.org/wiki/Application_programming_interface [29/04/2016].
Wikipedia (2016b). *Web API.* Accessed at: https://en.wikipedia.org/wiki/Web_API [29/04/2016].

Acknowledgements
The work described here is part of a wider team effort at Oxford University Press. In particular, I would like to thank the following members of LEAP engineering team in the Dictionary Technology Group: Artemis Parvizi, Edward Smith, Matt Kohl, and Meritxell Gonzàlez Bermúdez.