API-Powered Dictionary Websites

Sandro Cirulli

Oxford University Press

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 two African languages websites for isiZulu and Northern Sotho. The backend of these websites is based on an API retrieving data in RDF from a triple store and delivering data to the frontend in JSON-LD.

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.

1 Introduction

At the end of 2014 Oxford University Press (OUP) launched the Oxford Global Language (OGL) initiative [5] 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 to 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 [3] and Northern Sotho [4]. The backend of these websites is based on an API (Application Programming Interface) retrieving data in RDF [7] from a triple store and delivering data to the frontend in JSON-LD [6].

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.

2 Technical Implementation

2.1 System Architecture

Figure 1 shows the production system architecture for the isiZulu and Northern Sotho websites. 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 websites backend, including the API, the application server, and the data store

For the frontend and the security layers OUP relies on 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 for interacting with the data, and a GraphDB triple store for storing both OUP language data and user contributed data in RDF; the whole backend infrastructure runs on Amazon EC2 instances [1] using a microservices architecture based on Docker [2].

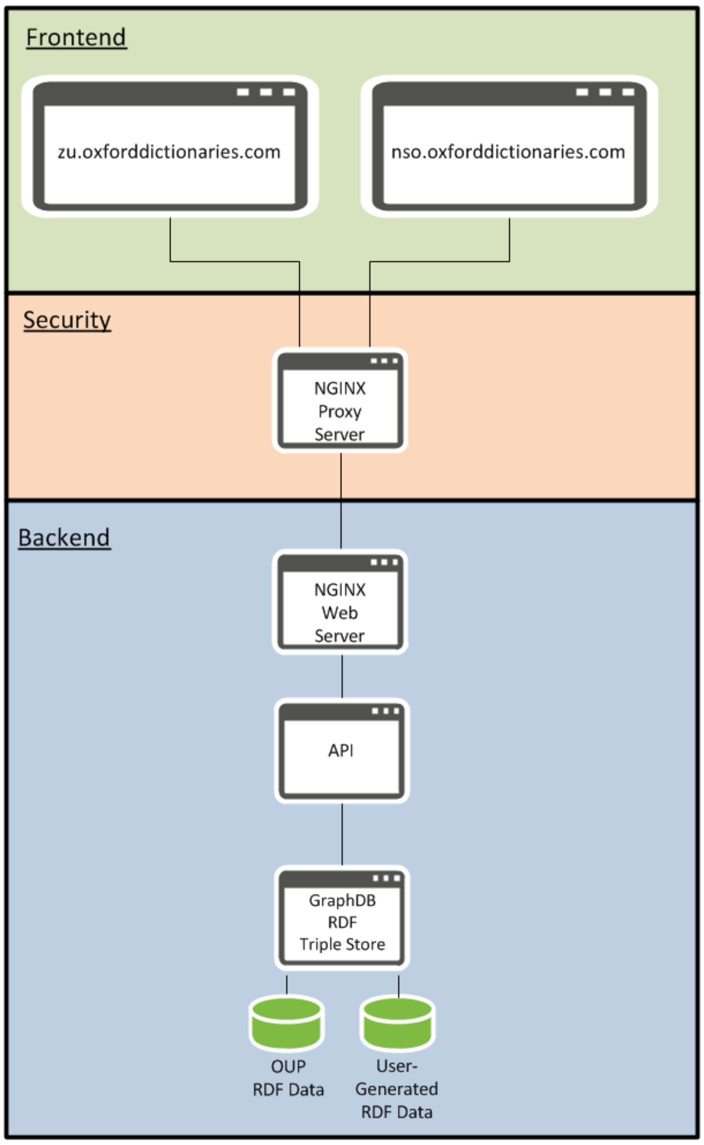


Figure 1: Production Architecture

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 [8] [9]. 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) and returns the HTTP status code of the SPARQL query to the API (1.5). The HTTP status code is mapped and transmitted back to the frontend (1.6) which displays a confirmation message to the user (1.7). The new entry created by the user is stored on the triple store 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 SPARQL query and send it to the triple store (2.3). The triple store runs the SPARQL query (2.4) and returns the results in RDF to the API (2.5). The API serializes the RDF into JSON-LD and returns it to the frontend (2.6). Finally, the website displays the entry to the user in an HTML page (2.7).

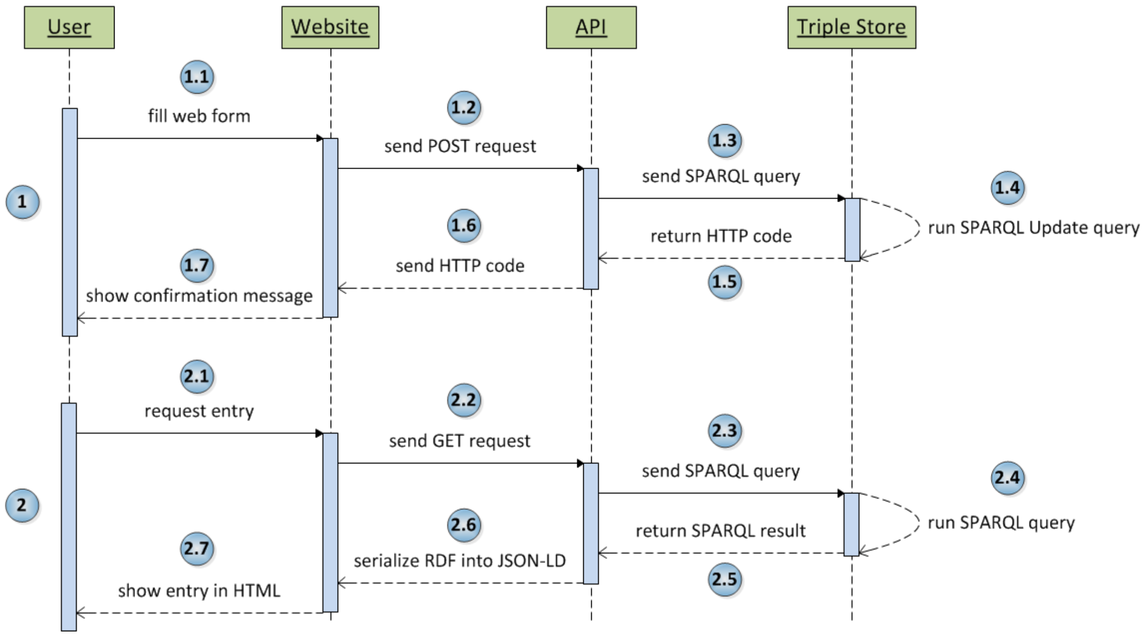
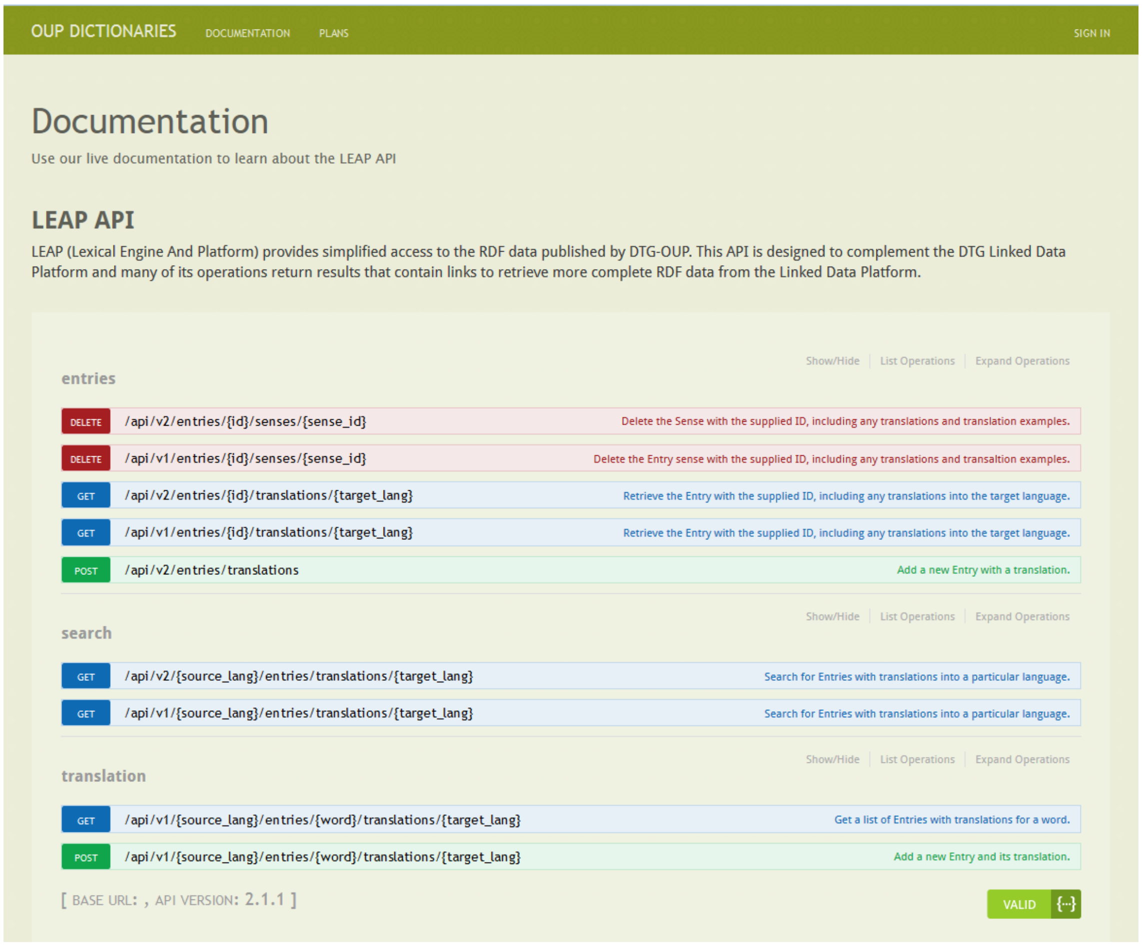


Figure 2: API GET and POST requests

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 are similar to those illustrated in Figure 2.

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


OUP DICTIONARIES 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/v2/entries/{id}/senses/{sense_id}	Delete the Sense with the supplied ID, including any translations and translation examples.
DELETE	/api/v1/entries/{id}/senses/{sense_id}	Delete the Entry sense with the supplied ID, including any translations and transaltion examples.
GET	/api/v2/entries/{id}/translations/{target_lang}	Retrieve the Entry with the supplied ID, including any translations into the target language.
GET	/api/v1/entries/{id}/translations/{target_lang}	Retrieve the Entry with the supplied ID, including any translations into the target language.
POST	/api/v2/entries/translations	Add a new Entry with a translation.

search

Show/Hide | List Operations | Expand Operations

GET	/api/v2/{source_lang}/entries/translations/{target_lang}	Search for Entries with translations into a particular language.
GET	/api/v1/{source_lang}/entries/translations/{target_lang}	Search for Entries with translations into a particular language.

translation

Show/Hide | List Operations | Expand Operations

GET	/api/v1/{source_lang}/entries/{word}/translations/{target_lang}	Get a list of Entries with translations for a word.
POST	/api/v1/{source_lang}/entries/{word}/translations/{target_lang}	Add a new Entry and its translation.

[BASE URL: , API VERSION: 2.1.1]

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 showed 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

The OGL initiative aims to develop several dictionaries and languages resources especially for digitally under-represented languages in the next years. OUP is currently developing the second phase of its programme and plans to launch in 2016 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 Hardware/Software Requirements

Apart from a projector and Internet access at a decent speed, no specific hardware or software requirements is needed to carry out the software presentation.

References

[1] Amazon Web Services, Inc. *Amazon EC2*. https://aws.amazon.com/ec2. Accessed: November 5, 2015.

[2] Docker. *Docker - Build, Ship, Run Any App, Anywhere*. https://www.docker.com. Accessed: November 5, 2015.

[3] Oxford University Press. *Explore & Translate isiZulu - Oxford Dictionaries*. https://zu.oxforddictionaries.com. Accessed: November 5, 2015.

[4] Oxford University Press. *Explore & Translate Northern Sotho - Oxford Dictionaries*. https://nso.oxforddictionaries.com. Accessed: November 5, 2015.

[5] Oxford University Press. *Oxford's Global Language Initiative*. http://www.oxforddictionaries.com/words/oxfordlanguages. Accessed: November 5, 2015.

[6] W3C. *JSON-LD 1.0*. http://w3.org/TR/json-ld. Accessed: November 5, 2015.

[7] W3C. *Resource Description Framework (RDF)*. http://www.w3.org/RDF. Accessed: November 5, 2015.

[8] Wikipedia. *Application Programming Interface*. https://en.wikipedia.org/wiki/Application_programming_interface. Accessed: November 5, 2015.

[9] Wikipedia. *Web API*. `https://en.wikipedia.org/wiki/Web_API`. Accessed: November 5, 2015.