Bioinformatics, 38(7), 2022, 2077–2079
https://doi.org/10.1093/bioinformatics/btac017
Advance Access Publication Date: 10 January 2022
Applications Note

Databases and ontologies

BioThings SDK: a toolkit for building high-performance data APIs in biomedical research

Sebastien Lelong, Xinghua Zhou, Cyrus Afrasiabi, Zhongchao Qian, Marco Alvarado Cano, Ginger Tsueng, Jiwen Xin, Julia Mullen, Yao Yao, Ricardo Avila, Greg Taylor, Andrew I. Su and Chunlei Wu *

Department of Integrative Structural and Computational Biology, The Scripps Research Institute, La Jolla, CA 92037, USA

*To whom correspondence should be addressed.

Associate Editor: Zhiyong Lu

Received on October 20, 2021; revised on December 10, 2021; editorial decision on January 4, 2022; accepted on January 8, 2022

Abstract

Summary: To meet the increased need of making biomedical resources more accessible and reusable, Web Application Programming Interfaces (APIs) or web services have become a common way to disseminate knowledge sources. The BioThings APIs are a collection of high-performance, scalable, annotation as a service APIs that automate the integration of biological annotations from disparate data sources. This collection of APIs currently includes MyGene.info, MyVariant.info and MyChem.info for integrating annotations on genes, variants and chemical compounds, respectively. These APIs are used by both individual researchers and application developers to simplify the process of annotation retrieval and identifier mapping. Here, we describe the BioThings Software Development Kit (SDK), a generalizable and reusable toolkit for integrating data from multiple disparate data sources and creating high-performance APIs. This toolkit allows users to easily create their own BioThings APIs for any data type of interest to them, as well as keep APIs up-to-date with their underlying data sources.

Availability and implementation: The BioThings SDK is built in Python and released via PyPI (https://pypi.org/project/biothings/). Its source code is hosted at its github repository (https://github.com/biothings/biothings.api).

Contact: cwu@scripps.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Concerns for reproducibility and the ethical considerations in ownership of publicly funded research has led to an increased effort to make biomedical research papers and data more accessible. While the open access/open data movement made biomedical research data more accessible, it did not enhance its re-usability. The FAIR data guiding principles were formed to address some of the issues and limitations of the open data movement (Wilkinson *et al.*, 2016). For data to be both accessible, interoperable and reusable, metadata standardization is needed. General online repositories like Figshare or Zenodo impose basic metadata requirements which help to standardize some basic metadata, but a lot of biomedical databases and repositories utilize their own metadata schema (European Organization for Nuclear Research, 2013; Singh, 2011). In addition to existing databases and repositories, web-based Application Programming Interfaces (APIs) play an important role in making biomedical research data more accessible and reusable and often have their own fragmented metadata.

We previously created high-performance, scalable, annotation as a service APIs that aggregated and provided gene and variant metadata (i.e. annotations) (Xin *et al.*, 2016): MyGene.info (Xin *et al.*, 2010) and MyVariant.info (Xin *et al.*, 2014). These APIs aggregate identifiers and other metadata from gene-specific or variant-specific resources and serve up these annotations as a RESTful service effectively increasing the interoperability of data from various gene and variant specific resources. This allows users to address two key issues in bioinformatics pipelines that make keeping data up-to-date difficult: (i) data storage, (ii) data fragmentation (i.e. downloading, tracking and updating data, versus constantly updating parsers for multiple resources). Our team has since built similar APIs for chemical entities [MyChem.info (Lelong *et al.*, 2015)], disease entities [MyDisease.info (Xin *et al.*, 2015)] and taxonomy [t.biothings.io/ (Lelong *et al.*, 2017)]. These APIs are collectively referred to as the BioThings APIs. Calling these BioThings APIs enable users to quickly perform id conversion and knowledge retrieval for ease of querying the respective data resources. By relying on Annotation as a service APIs, users reduce the need to download and store data from multiple resources, track and update said data locally, and resolve changes from multiple resources. This enhances the accessibility, interoperability and reusability of data scattered across multiple resources.

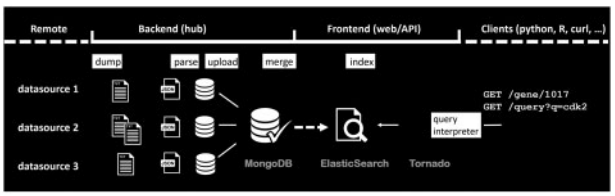

Fig. 1. The overall architecture of the BioThings SDK. Its data hub (backend) component handles data source monitoring, parsing/uploading and then merging across all data sources. Its web API (frontend) component handles data indexing and processes user queries

The BioThings APIs were built with architecture demonstrated to be flexible and scalable. We currently have APIs that cover five different types of biomedical data out of the potentially hundreds if not thousands of biomedical entity types. To increase the FAIRness of different biomedical entity types, we encourage and empower other research groups to create and share their own APIs using the BioThings architecture. We reviewed the process of creating BioThings API, compared the architecture and applied the best practices and lessons learned from the process to create a python-based Software Development Kit (SDK). The SDK allows users to create a high-performance API once the user provides a data source parser. Once the API is created, the BioThings API client can be adapted for use with the API. In addition, the BioThings SDK includes a data source management console (i.e. the dashboard) which helps to handle data source monitoring and download scheduling (tasks important for keeping aggregated data up-to-date). Overall, we provide BioThings SDK as a system to rapidly increase the availability of high-quality, sustainable production-ready APIs to the research community.

2 Implementation

The SDK consists of a data hub component and a web API component (Fig. 1). The data hub processes each data source ('dump') and stores the parsed JSON objects ('upload') in MongoDB (MongoDB, 2021). All JSON objects can then be merged across data sources (based on the same primary keys) and indexed in Elasticsearch (Tong, 2013). The use of Elasticsearch ensures scalability allowing easy addition of nodes to handle increasing volumes of data and queries (Elastic.co, 2021). The web API handles the user's Elasticsearch queries and returns back the query results. The API is built with the Python-based Tornado web framework (Tornadoweb.org, 2021), which is non-blocking and suitable for implementing high-performance APIs. Both the tornado web servers and the Elasticsearch cluster can be individually scaled up on demand as the incoming requests or data scale increase. In theory, the maximum expected throughput is limited by the capacity of the cloud-providers (like AWS), not the BioThings SDK itself.

The data hub also includes a set of utilities to monitor the data source files and triggers the automated data-downloads data-parsing and releasing. This feature greatly simplifies the data-processing burden and keeps the data always up-to-date. The SDK is regularly released as a package via PyPI and its source code is hosted at the github repository. Detailed documentation for the BioThings SDK (https://docs.biothings.io), a full tutorial, and preconfigured Dockerized development environment and a web interface (called 'BioThings Studio') are also provided (https://docs.biothings.io/en/latest/tutorial/studio.html#tutorial).

The minimal hardware requirements for setting up an API using BioThings SDK is moderate (>8 GB memory and >10 GB storage) and can be run on an average laptop for the development. Optimally, computing power would be scaled in accordance with the amount of data processed.

3 Results

To create a RESTful API out of a data source (or add a data source into an existing API), a 'data plugin' must be written. The data plugin will need to follow a simple but specific architecture consisting of three primary python modules: the dumper.py, which tells the system where, how, and even how frequently to get the data; the parser.py, which tells the system how to parse and transform the data; and the upload.py, which tells the system how to load and merge the data into MongoDB. The BioThings SDK comes packaged with many helper functions which may be called in the data plugin files to reduce the amount of coding needed for common or redundant actions. For example, the SDK has helpers for downloading data via ftp, http, git, google drive and more that can be used in the data plugin dumper script. In most straightforward scenarios, a data plugin can include just a parser.py and a manifest.json file with sufficient metadata (Lelong, 2020a), however, with dedicated dumper.py and upload.py, users have additional fine-tuned controls to customize the desired behavior (Lelong, 2020b). Once the data plugin has been created, it can be registered within the dashboard. Additional examples on writing plugins can be found in the aforementioned full tutorial.

Registering the plugin within the dashboard allows the operator to manually trigger or schedule data updates and monitor the plugin for issues with uploading, merging and indexing (Supplementary Fig. S1). If the SDK is used to create an API for a single data source, each upload for the data into MongoDB can be considered a single build and can be configured as such. If there are multiple data sources, the build can be configured to handle the upload and merging of the data sources. Unique identifiers are needed for the Elasticsearch index, and the SDK will merge data from different sources whenever duplicate identifiers are found. For example, if a BioThings API like MyGene.info consisted of only a single data source (NCBI Gene), then each upload from the data plugin would have only the annotations available from NCBI entrez and each build would consist only of updates from NCBI Gene. In actuality MyGene.info provides gene-specific annotations from NCBI Gene, Ensembl, Uniprot, NetAffx, PharmGKB, UCSC, CPDB, ClinGen, REACTOME and more. Since each data source has a different frequency with which updates are released, the data plugins are scheduled to update according to the releases. By mapping the annotations from each source to NCBI gene or Ensembl ids, any update will trigger a new merge and build with each JSON object in the database containing annotation data from all relevant data sources. Build configurations can be created and edited directly from the dashboard (Supplementary Fig. S2).

Once the build has been completed, it will be sent to Elasticsearch for indexing. Elasticsearch can be customized to map/index specific fields/properties and this customization allows for the creation of customized API queries. Fields or properties not indexed by Elasticsearch will still exist within the data but cannot be queried. The SDK dashboard also has analysis tools for inspecting the data to automatically generate a potential mapping file for Elasticsearch. After the build is indexed by Elasticsearch, a new API can be created from within the dashboard (Supplementary Fig. S3).

4 Conclusions

We have created an SDK for building RESTful APIs for the biomedical research community. The BioThings SDK has been used to build over 50 APIs which collectively includes and exposes over 1.7 billion records (Supplementary Table S1). The SDK is flexible enough to create APIs for metadata (annotations), data, ontologies and more. Though the usage of these APIs remains largely untracked, the average monthly requests for the 5 APIs where usage was tracked exceeded 53 million from over 20 000 unique IPs. In addition, the various APIs serve data to other tools, projects and resources such as the Monarch Initiative, CIViC, MyGene2, Open Humans and more. The BioThings SDK comes with a dashboard that allows the operator to easily manage some of the more redundant data wrangling tasks. Although only Python is currently supported, we are currently exploring options for supporting other languages in the future.

Funding

This work was supported by the US National Institute of Health (https://www.nih.gov/) [OT2TR003445 to C.W., R01GM083924 to A.I.S. and C.W.]. The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

Conflict of Interest: none declared.

References

Elastic.co. (2021) *Scalability and resilience: clusters, nodes, and shards | Elasticsearch Guide [7.15] | Elastic*. https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html (29 November 2021, date last accessed).

European Organization for Nuclear Research. (2013) Zenodo. CERN. https://doi.org/10.25495/7gxk-rd71.

Lelong,S. *et al.* (2015) *MyChem.info | Chemical and Drug Annotation as a Service*. MyChem.info. http://mychem.info/ (3 August 2021, date last accessed).

Lelong,S. *et al.* (2017) *BioThings Taxonomy API*. https://t.biothings.io/ (1 October 2021, date last accessed).

Lelong,S. (2020a) *MVCGI BioThings Studio data plugin demo*. https://github.com/sirloon/mvcgi (1 October 2021, date last accessed).

Lelong,S. (2020b) *MVCGI Advanced BioThings Studio data plugin demo*. https://github.com/sirloon/mvcgi_advanced (1 October 2021, date last accessed).

MongoDB. (2021) What Is MongoDB? https://www.mongodb.com/what-is-mongodb (3 August 2021, date last accessed).

Singh,J. (2011) FigShare. *J. Pharmacol. Pharmacother.*, **2**, 138–139.

Tong,Z. (2013) *What is an Elasticsearch Index? | Elastic*. Elastic.co. https://www.elastic.co/blog/what-is-an-elasticsearch-index (3 August 2021, date last accessed).

Tornadoweb.org. (2021) *Tornado Web Server – Tornado 6.1 documentation*. https://www.tornadoweb.org/en/stable/ (3 August 2021, date last accessed).

Wilkinson,M.D. *et al.* (2016) The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data*, **3**, 160018.

Xin,J. *et al.* (2010) *MyGene.info | Gene Annotation as a Service*. MyGene.info. http://mygene.info/ (3 August 2021, date last accessed).

Xin,J. *et al.* (2014) *MyVariant.info | Variant Annotation as a Service*. MyVariant.info. http://myvariant.info/ (3 August 2021, date last accessed).

Xin,J. *et al.* (2015) *MyDisease.info | Disease Annotation as a Service*. MyDisease.info. http://mydisease.info/ (3 August 2021, date last accessed).

Xin,J. *et al.* (2016) High-performance web services for querying gene and variant annotation. *Genome Biol.*, **17**, 91.