Process-Level Integration for Linked Open Data Development Workflows: A Case Study

Manuel Fiorelli [0000-0001-7079-8941] [1], Armando Stellato [0000-0001-5374-2807] [1], Ilaria Rosati [0000-0003-3422-7230] [2], Nicola Fiore [0000-0002-9538-2966] [3]

[1] University of Rome Tor Vergata, Department of Enterprise Engineering, via del Politecnico 1, 00133 Roma, Italy
{manuel.fiorelli,stellato}@uniroma2.it

[2] Institute of Research on Terrestrial Ecosystems (IRET), National Research Council (CNR), 73100 Lecce, Italy
ilaria.rosati@cnr.it

[3] LifeWatch ERIC Service Centre, Lecce, Italy
nicola.fiore@lifewatch.eu

Abstract. Dataset maintenance and development is a complex endeavor that necessitates the use of different systems through the dataset lifecycle. The semantic web succeeded in setting common standards that enable data interchange between these systems. Still, there is often a lack of interoperability and integration at the process level, indispensable to use different systems in a single, combined workflow. In this regard, we considered use cases concerning the interaction of dataset editors with content publication/fruition systems, which provide a linked data interface, and dataset catalogs, which meet the FAIR principle of findability and persistency. As a case study, we considered ShowVoc and OntoPortal as examples of these two classes of systems. Having committed to the collaborative knowledge editor VocBench 3, we contributed extensions for it that address the defined use cases. Evaluating our contributions, we identified some aspects of future improvement in the relevant subset of VocBench 3.

Keywords: linked open data, workflow, integration, catalogs, publication, OntoPortal, ShowVoc, VocBench

1 Introduction

The semantic web [1] was conceived as an extension of the traditional document web, aimed at enabling machines to better understand and process resources on the web through the explication of their intended semantics. Although the original vision has not been delivered yet, research on semantic web – renewed by the linked open data paradigm [2] – succeeded in the definition of common standards for data publication, reuse, and integration [3], towards the realization of a web of data, evolving the current document web into a global data space [4]. Unsurprisingly, a lot of systems have been developed to cover the different stages of the data lifecycle on the semantic web, including – but not limited to – (collaborative) editing, linking, and publication. Related to the latter, the semantic web has traditionally relied on decentralization and, with the advent of link open data, data publishers are in charge of setting up HTTP servers so that resource IRIs resolve into different representations (e.g., HTML for humans and different RDF serializations for machines), while the primary mechanism to find information is link traversal – often called "follow your nose". Search engines (e.g., Google, Bing, Yahoo, Baidu), which have become the entry point to the web, are still not widely established on the semantic web (nor have existing actors provided large support for it, having developed their own internal "knowledge graphs"); nonetheless, some forms of centralization have already made their way to the semantic web through data catalogs [5,6,7], which address a variety of needs, not least persistency against failure of the original data sources. In fact, catalogs acquire primary importance at the intersection of the semantic web and the open data movement. The latter has promoted data accessibility as a means to increase accountability and transparency of governments and organizations as well to speed up scientific progress. The scientific open data community eventually formulated the FAIR principles [8] for data management and stewardship with the idea of reducing obstacles to data reuse. Metadata and catalogs play a critical role in this, as they facilitate the discovery of relevant datasets through keyword search and other criteria. Showing their closeness to the semantic web aims, the FAIR guidelines actually value machine-actionability, defined as "the capacity of computational systems to find, access, interoperate, and reuse data with none or minimal human intervention"[1]. As an example, metadata and catalogs have been used to orchestrate ontology matching processes [9,10].

Dataset development and publication according to the practices and principles just described require a complex workflow, which inevitably relies on different systems. The common standards set by the semantic web may support data interchange between these systems; still, there is often a need for better interoperability and integration at the process level – as required to use them in a single, combined workflow. In this regard, our contribution is a set of bridges between different systems that address data editing, publication/fruition, and cataloging.

In this study, we committed to VocBench 3 [11] as the editing system, and then exploited its extensible data loading and export facilities, as well as its support for dataset catalogs in order to enable interoperability with other relevant tools. As a case study, we developed concrete extensions for ShowVoc[2] – a new read-only companion to VocBench 3 for data publication/fruition – and OntoPortal[3] – a software for data catalogs originated from the BioPortal repository of biontologies. Thanks to our contributions, these two disparate systems have been successfully integrated into a cohesive ecosystem centered on VocBench 3.

The paper is structured as follows. Section 2 discusses related work. Section 3 defines our use cases. Section 4 describes our contribution in the context of VocBench 3 architecture. Section 5 describes the extensions that we developed for our case study. Section 6 is about evaluation. Section 7 contains the conclusions.

[1] `https://www.go-fair.org/fair-principles/`
[2] `http://showvoc.uniroma2.it/`
[3] `https://ontoportal.org/`

2 Related Work

There is a large number of editors for the semantic web addressing different concerns.

Protégé [12] is a renowned, long-standing open-source ontology editor. Extensibility is one of Protégé strong points, which contributed to its adoption as a de facto standard development platform for ontology-related research. WebProtégé [13] is a recent companion to the original software (now called Protégé Desktop) providing a collaborative, web-based ontology development environment. Generic ontology development tools may not be so effective to support dataset development conforming to specific modeling vocabularies, say SKOS(-XL) [14,15] for thesauri or OntoLex [16] for lexicons, requiring dedicated software [17,18,19].

PoolParty Semantic Suite [20] is a family of proprietary systems addressing data management, data ingestion – including acquisition form textual sources – and exploitation in semantic search. PoolParty features dedicated support for SKOS thesauri. TopQuadrant Enterprise Data Governance [21] is another propriety system with a strong focus on data governance, lineage-tracking, etc.

VocBench 3 is an open-source web-application for collaborative editing of ontologies, thesauri, lexicons, and RDF-datasets in general, complying with all relevant semantic web standards. Multi-model editing and compliance with standards are two tenants of VocBench 3 together with its several extensibility features (ranging from very complex plugins to implementations of predetermined extension points).

Most editing systems support data extraction from a variety of sources and loading the edited data onto diverse destinations. The latter includes publication of data. PoolParty integrates UnifiedViews [22], which can source raw data from downloads, HTTP API and PoolParty Concept Extraction, while allowing loading data onto a SPARQL endpoint. TopQuadrant Enterprise Data Governance can export data to files and, optionally, upload them to AWS S3 [23]. Both editors can export a dataset to a service for content fruition, respectively, TopBraid Explorer, and the Wiki Frontend (also supporting lightweight editing) or the Linked Data Frontend. A special use case of data input/output is related to tabular data (e.g., CSV files, spreadsheets, etc.) for which editing systems often have dedicated facilities [24]. As an example, VocBench 3 provides a dedicate tool, called Sheet2RDF [25], which builds upon the knowledge acquisition platform CODA [26].

Protégé has a plugin [27] to import data from BioPortal. It enables to reference entities found in ontologies hosted on BioPortal, and to import their definitions into the dataset being edited: indeed, bioontologies often combine concepts from different ontologies/namespaces. Similarly, PoolParty enables enriching a dataset with information fetched from (previously connected) Linked Data sources[4].

We conclude with a brief discussion of linked data publication software. Pubby[5] is the progenitor of a lineage of tools that implement HTTP resolution (requested by the

[4] https://help.poolparty.biz/en/user-guide-for-knowledge-engineers/advanced-features/linked-data-management---overview/linked-data-enrichment-with-poolparty.html

[5] https://github.com/cygri/pubby

linked data paradigm) on top of a SPARQL endpoint. More recent solutions include LodView[6] and Loddy[7], not to mention the ones integrated into editing tools or triple stores. Skosmos[8] combines traditional subject pages (i.e., the result of content negotiation) for SKOS thesauri with more sophisticated visualizations (e.g., alphabetic index over the dataset), search and multi-thesaurus management. ShowVoc extends this approach beyond thesauri to ontologies, lexicons, and datasets in general: it focuses on content fruition and cross-dataset operations (e.g., global search), and it also supports content negotiation.

3 Use Cases

We set the requirements for our contributions through the identification of the following use cases. They define different interaction scenarios between VocBench 3 – chosen as editing system – and dataset catalogs and content publication/fruition systems.

3.1 Pull

The first scenario concerns with the need to *pull* a dataset from a catalog. We can further distinguish between i) *loading data* and ii) *importing ontologies*, depending on whether the pulled data is being put in the editable part of the dataset or just available as read-only.

3.2 Pull-Push

The second scenario extends the previous one with the subsequent upload of the pulled data to another different system, using VocBench 3 as a pivot. This is probably a less common use case, dealing with the re-publication of already existing datasets.

3.3 Push-Push

The third scenario is related to an integrated workflow for dataset development. VocBench 3 enables collaborative editing of a dataset, which is then pushed to a system for data publication/fruition (e.g., ShowVoc) and another system for cataloging (e.g., an OntoPortal instance).

4 Architecture

VocBench 3 is very flexible in both exporting and loading data, thanks to extension points that allow different destinations and data sources to be linked, respectively.

[6] `https://github.com/LodLive/LodView`
[7] `https://bitbucket.org/art-uniroma2/loddy`
[8] `https://skosmos.org/`

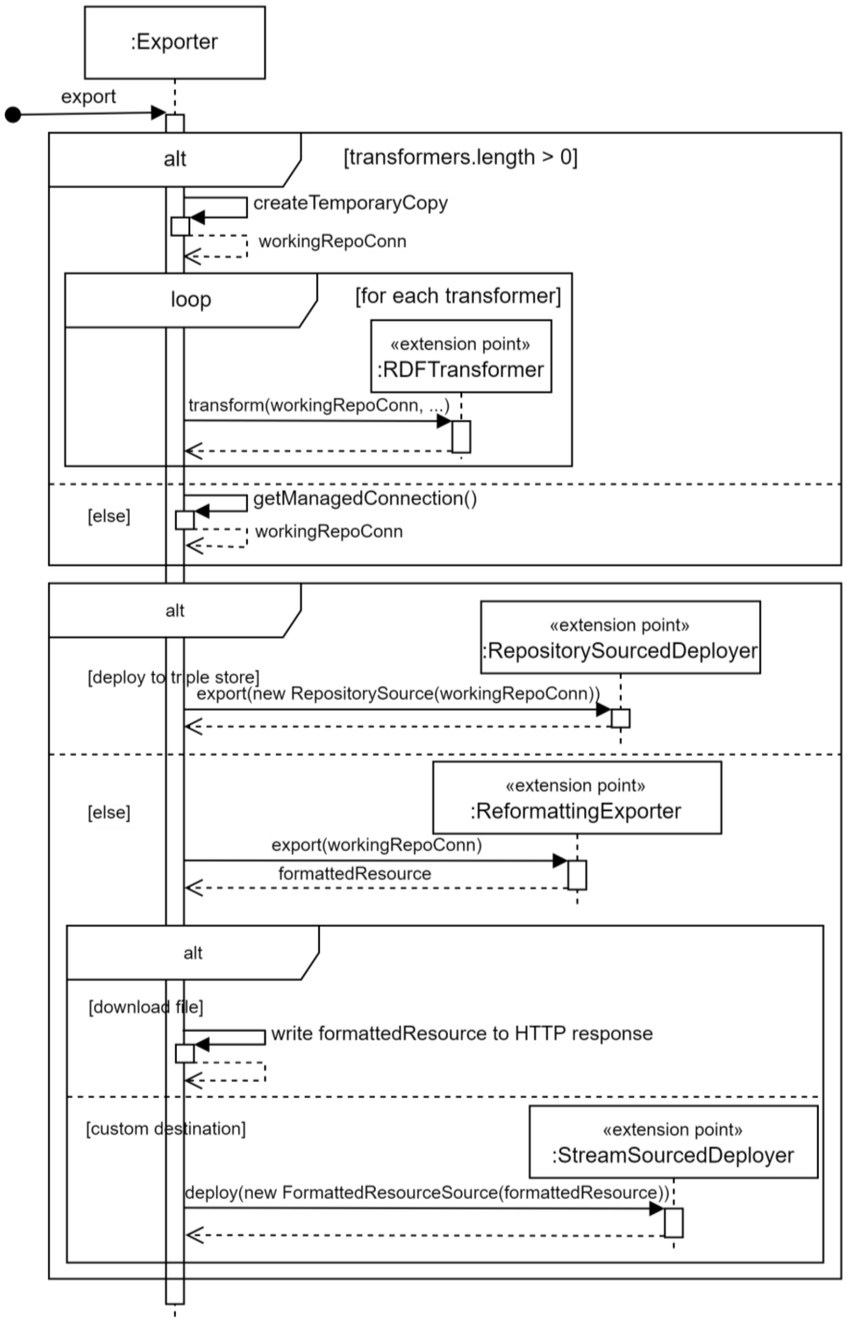


Fig. 1. Execution of the data export pipeline (UML2 sequence diagram)

Fig. 1 illustrates the export of a dataset to a i) a triple store, ii) a downloadable file, iii) a custom destination. In fact, the export chain is introduced by an optional chain of

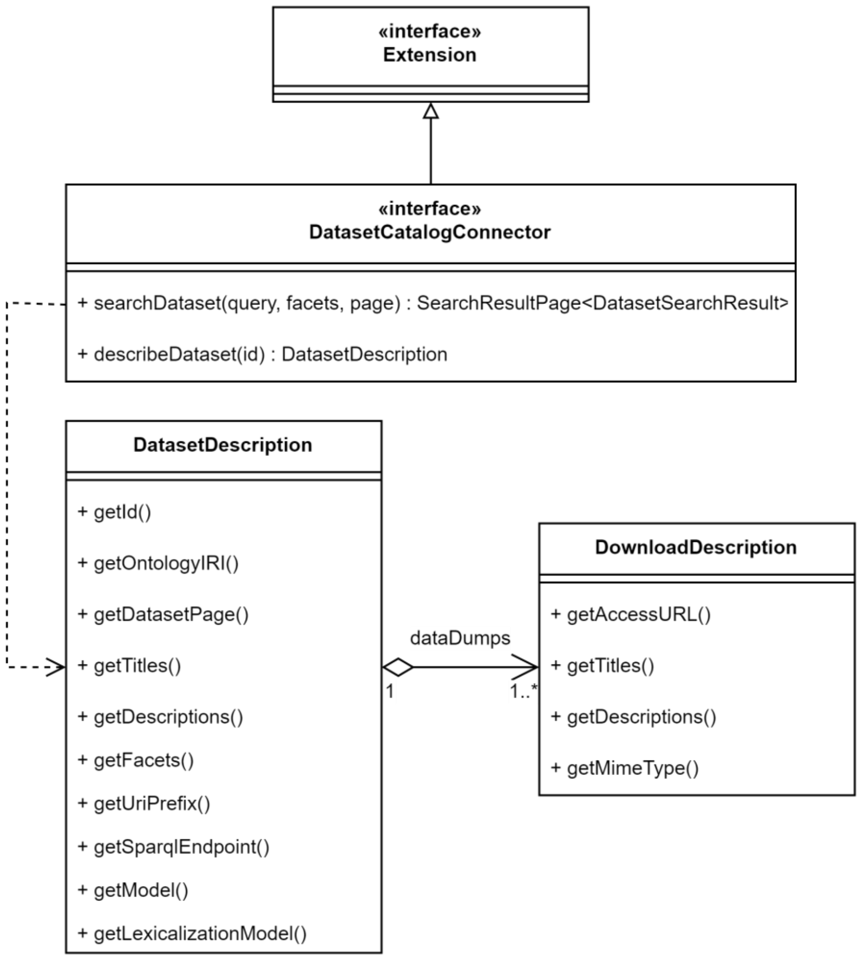
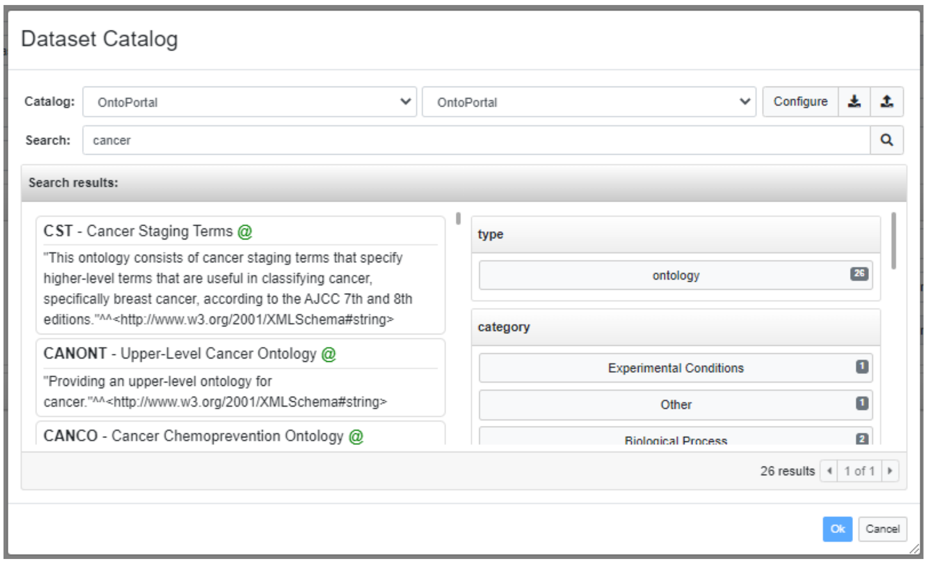


Fig. 2. Simplified UML class diagram for the extension point dataset catalog connector

RDF transformers that can process the data before being exported. These transformers are defined by an *extension point* that enables to plug diverse implementations.

Prepackaged transformers include one for simple property value update and another utilizing SPARQL to specify the transformation. These transformers are executed on a temporary copy of the dataset being exported, as they could perform potentially destructive changes that should not be applied to the original dataset. The rest of the export pipeline will use a working connection to this temporary copy; alternatively, when no transformation is necessary, no temporary copy is created, avoiding the (unnecessary) costs associated with that, while directly working with a (read-only) connection to the dataset being exported.


Dataset Catalog

Catalog: OntoPortal OntoPortal Configure

Search: cancer

Search results:

CST - Cancer Staging Terms @
"This ontology consists of cancer staging terms that specify higher-level terms that are useful in classifying cancer, specifically breast cancer, according to the AJCC 7th and 8th editions."^^<http://www.w3.org/2001/XMLSchema#string>

CANONT - Upper-Level Cancer Ontology @
"Providing an upper-level ontology for cancer."^^<http://www.w3.org/2001/XMLSchema#string>

CANCO - Cancer Chemoprevention Ontology @

type
ontology 26

category
Experimental Conditions 1
Other 1
Biological Process 2

26 results 1 of 1

Ok Cancel

Fig. 3. Dataset catalog user interface

The export pipeline continues differently depending on the destination type. Unless the exported data is to be downloaded, a *deployer* is invoked ultimately. Again, this is an extension point, which is further specialized into *repository sourced deployer* (for deploying to a triple store) and *stream sourced deployer* (for deploying to a stream-oriented destination). In this context, triple store is broadly intended as any destination which can be fed with RDF data without the need for an explicit conversion beforehand. Beyond actual triple stores, this category includes RDF data catalogs and RDF data publication services. In other scenarios, a *reformatting exporter* can be used to explicitly serialize the (possibly transformed) data into a byte stream conforming to some data format.

Data loading is somehow symmetric to the export mechanism described so far. Unless data is uploaded by the user when starting the process, a *loader* – another extension point – is used to pull data from a triple store (i.e., *repository targeting loader*) or a stream-oriented source (i.e., *stream targeting loader*). Unless directly loading RDF data from a triple store, it is necessary to use an *RDF lifter* – another extension point – to convert the pulled byte stream into actual RDF data. At the end of the loading pipeline, a transformation chain can be used to implement data massaging.

Another requirement lifted from the use cases is the need for accessing a data catalog to lookup for datasets. We addressed it using the extension point *dataset catalog connector* (see **Fig. 2**). A connector provides an operation to *search for a dataset* (returning a paginated list of results) and an operation to *obtain a description of a dataset*. The latter is modeled with a class that includes general metadata (e.g., titles and descriptions) mostly targeted at humans, as well as other more machine actionable metadata, such as *model* (telling the difference between an OWL ontology, SKOS thesaurus, etc.), *lexicalization model* (telling how lexical information is represented), and the *data dumps*. These are described in turn with actionable metadata such as the *access URL* (to fetch the dump) and *MIME-type* (telling the data format and charset).

Data catalogs are presented as another option for importing an ontology or loading data. In fact, they just provide the ontology IRI or the data dump access URL to the already existing machinery for ontology import – from an (alternative) URL – or data loading – from an URL. **Fig. 3** illustrates the dialog for searching a dataset on a catalog. The first step is to select a connector implementation (e.g., the contributed connector for OntoPortal) and, if required, configure it. At this point, it is possible to issue a query based on some search terms. The results are presented as a list, complemented on the right by catalog-specific facets to refine the query. When a dataset is chosen, its description is shown on the right.

5 Case Study: ShowVoc and OntoPortal

Our case study for the architecture discussed in Section 4 is the use of VocBench 3 to integrate (at the process level) both ShowVoc (for data publication and fruition) and OntoPortal (for data cataloging).

Looking at use cases in Section 3, all but the pull scenario require the ability to deploy data to either system. We thus implemented a *deployer* for each application, which have been considered as *triple store destinations.* The ShowVoc deployer works by communicating with the backend server (Semantic Turkey [28]) of ShowVoc, while the OntoPortal deployer uses the REST API originally developed by BioPortal.

In the pull scenario, we need to look up a dataset in one of these applications. To this end, we implemented a *dataset catalog connector* for each system.

6 Evaluation

We compiled **Table 1** to ascertain that our contributions satisfy the needs associated with the scenarios discussed in Section 3.

The response time of dataset catalog connectors and deployers is mostly determined by the performance of the backing service. This holds true, in particular, when the extension is implemented through a single request to the backing service with a minimal overhead. We consider uninformative any performance discussion in this case, since there is little that can be ascribed to our implementation. In fact, the OntoPortal connector does not fall into such scenario, since the OntoPortal API does not have a dedicated ontology search endpoint – while offering some for concept-lookup.

We followed the example of the OntoPortal web application, which implements a searchable catalog in the *ontologies* page. In absence of a dedicated search API, the web application first retrieves all ontologies (metadata), and all categories and groups, which can be used to classify ontologies: the description of an ontology just references the IRI of a group or category, thus necessitating the description of the latter to obtain their human-friendly names. In fact, the description of an ontology does not contain metadata used (in the user interface) as search facets (i.e., modification date,

Table 1. Requirements traceability matrix

Scenario	Features
pull – i) loading data	use of data catalogs in data loading, or in pre-loading when creating a project
pull – ii) importing ontologies	use of data catalogs when importing an ontology
pull - push	those in the first row, together with the use of deployers to "triple stores"
push - push	use of deployers to "triple stores"

data format), nor does it include the textual description of the ontology. These missing attributes are, indeed, derived from the latest submission for the ontology. Consequently, the web application also fetches the latest submissions for all ontologies (through a single request). All the retrieved information is then incorporated into the *ontologies* page, which implements search completely client-side (without further requests to the backend).

Given the complexity of the implementation of the search functionality for OntoPortal, it is worth evaluating its performance through some experiments (see **Table 2**). We made these experiments against BioPortal, which is the largest installation of OntoPortal to date. The first two rows describe the (minimum, maximum, and average) response time of a search using our dataset catalog connector with the term "cancer". The difference between these two is whether we perform the additional step of retrieving the list of all latest submissions. The third row is the time required to download just the HTML of the *ontologies* pages (which embeds all information for doing a search). The fourth row is the time required to retrieve all ontologies using the API (including just the required attributes in the obtained objects).

Our measures are consistent with the fact that the third and fourth rows should be an upper and lower bound of our connector runtime, respectively: without a search API it is not possible to be quicker than a listing of all ontologies, and the implementation should not be slower than the ontologies page, which contains all required information. The difference between the first and second row is 1.2 seconds on average, justifying our decision to let users decide which configuration to use.

We conclude the evaluation with a discussion of the VocBench 3 architecture, highlighting some aspects that could require future improvements.

Dataset catalog connectors must be stateless (by design); therefore, current service processing cannot benefit from the work done for previous requests. This is not a problem when the actual work is done by the connected service. Conversely, the OntoPortal connector implements the search locally, against metadata about all ontologies retrieved on the fly. In this scenario, it makes sense to enable storing this metadata at least on the session, avoiding the need to repeat the costly API invocation that would most likely return exactly the same results in the short term. The problem with caching as obvious is the strategy to evict the cache, in order to guarantee the freshness of the result.

As discussed about the VocBench 3 architecture, dataset catalog connectors are loosely coupled to the data loading and ontology import machinery as a mere provider of the data access URL. However, this could be problematic when the location of a

Table 2. Response time evaluation related to the OntoPortal connector (using 50 repetitions)

	Min (s)	**Max (s)**	**Avg (s)**
OntoPortal Dataset Catalog connector w/o submissions	2.306785	3.182033	2.425709
OntoPortal Dataset Catalog connector w/ submissions	3.364919	8.696316	3.59604518
BioPortal "ontologies" page	3.365554	5.761266	3.752337
BioPortal API ontologies collection	1.232167	2.841468	1.557863

resource is not sufficient. For example, when accessing a protected resource, we need to authorize the data access. For OntoPortal, we addressed this use case, by embedding the API key as a query parameter – a controversial practice, which is also used in some parts of the OntoPortal web application. A possible solution is to let the data loading and ontology import machinery to directly invoke the dataset catalog connector when downloading the data, in order to delegate to it authentication concerns.

7 Conclusions

Dataset development for the semantic web unavoidably requires the use of different tools, which have to be integrated into a unified workflow at the process level. Assigning a pivotal role to the editing environment, we committed to the use of VocBench 3 and identified some use cases concerning its interaction with dataset catalogs and fruition/publication systems. We considered an actual case study using OntoPortal and ShowVoc as concrete examples of these two categories of systems. We addressed the defined use cases through extensions to VocBench 3 related to data load and export processes, and the support for dataset catalogs. The evaluation of our contributions then allowed us to identify possible future improvements to the VocBench 3 architecture.

Acknowledgements

This work has been supported by the European e-science infrastructure for biodiversity and ecosystem research LifeWatch Eric and by LifeWatch Italy through the project "LifeWatchPLUS" - PIR01_00028. It also received support from the KATY project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017453.

References

1. Berners-Lee, T., Hendler, J.A., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities.

Scientific American 284(5), 34-43 (2001)

2. Berners-Lee, T.: Linked Data. In: Design Issues (2006). https://www.w3.org/DesignIssues/LinkedData.html
3. Shadbolt, N., Berners-Lee, T., Hall, W.: The Semantic Web Revisited. IEEE Intelligent Systems 21(3), 96-101 (2006)
4. Heath, T., Bizer, C.: Linked data: Evolving the web into a global data space. Synthesis Lectures on the Semantic Web: Theory and Technology 1(1), 1-136 (2011)
5. Vandenbussche, P.-Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web. Semantic Web 8(3), 437-452 (2017)
6. Whetzel, P.L. et al.: BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Research 39(suppl_2) (2011)
7. Jackson, R. et al.: OBO Foundry in 2021: operationalizing open data principles to evaluate ontologies. Database 2021 (2021) baab069.
8. Wilkinson, M.D. et al.: The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3(160018) (2016)
9. Fiorelli, M. et al.: Metadata-driven Semantic Coordination. In: Garoufallou, E., Fallucchi, F., De Luca, E.W. (eds) Metadata and Semantic Research. 13th International Conference, MTSR 2019, Rome, Italy, October 28–31, 2019, Revised Selected Papers. Communications in Computer and Information Science, vol. 1057. Springer, Cham (2019)
10. Mochol, M., Jentzsch, A.: Towards a Rule-Based Matcher Selection. In: Gangemi, A., Euzenat, J. (eds) Knowledge Engineering: Practice and Patterns. EKAW 2008. Lecture Notes in Computer Science, vol. 5268. Springer, Berlin, Heidelberg (2008), pp.109-119
11. Stellato, A. et al.: VocBench 3: A collaborative Semantic Web editor for ontologies, thesauri and lexicons. Semantic Web 11(5), 855-881 (2020)
12. Musen, M.A.: The Protégé Project: A Look Back and a Look Forward. AI Matters 1(4), 4-12 (2015)
13. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtégé: A Collaborative Ontology Editor and Knowledge Acquisition Tool for the Web. Semantic Web 4(1), 89-99 (2013)
14. World Wide Web Consortium (W3C): SKOS Simple Knowledge Organization System Reference. In: World Wide Web Consortium (W3C) (August 18, 2009). http://www.w3.org/TR/skos-reference/. Accessed 22 March 2011
15. World Wide Web Consortium (W3C): SKOS Simple Knowledge Organization System eXtension for Labels (SKOS-XL). In: World Wide Web Consortium (W3C) (August 18, 2009). http://www.w3.org/TR/skos-reference/skos-xl.html. Accessed 22 March 2011
16. Cimiano, P., McCrae, J.P., Buitelaar, P.: Lexicon Model for Ontologies: Community Report, 10 May 2016. Community Report, W3C (2016) https://www.w3.org/2016/05/ontolex/
17. Jupp, S., Bechhofer, S., Stevens, R.: A Flexible API and Editor for SKOS. In: Aroyo, L. et al. (eds) The Semantic Web: Research and Applications. Lecture Notes in Computer Science, vol. 5554. Springer, Berlin, Heidelberg (2009), pp.506-520
18. Fiorelli, M., Pazienza, M.T., Stellato, A.: Semantic turkey goes SKOS managing

knowledge organization systems. In: I-SEMANTICS '12 Proceedings of the 8th International Conference on Semantic Systems, Graz, Austria, pp.64-71 (2012)

19. Mochón, G., Méndez, E.M., Bueno de la Fuente, G.: 27 pawns ready for action: A multi-indicator methodology and evaluation of thesaurus management tools from a LOD perspective. Library Hi Tech 35(1), 99-119 (2017)
20. PoolParty Semantic Suite - Semantic Technology Platform. https://www.poolparty.biz/
21. TopBraid Enterprise Data Governance. https://www.topquadrant.com/products/topbraid-enterprise-data-governance/
22. Knap, T. et al.: UnifiedViews: An ETL tool for RDF data management. Semantic Web 9(5), 661-676 (2018)
23. AWS: Amazon S3. In: Amazon Web Services. https://aws.amazon.com/s3/
24. Fiorelli, M., Stellato, A.: Lifting Tabular Data to RDF: A Survey. In: Garoufallou, E., Ovalle-Perandones, M.-A. (eds) Metadata and Semantic Research. MTSR 2020. Communications in Computer and Information Science, vol. 1355. Springer, Cham (2021), pp.85-96
25. Fiorelli, M., Lorenzetti, T., Pazienza, M.T., Stellato, A., Turbati, A.: Sheet2RDF: a Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF. In: Ali, M., Kwon, Y.S., Lee, C.-H., Kim, J., Kim, Y. (eds) Current Approaches in Applied Artificial Intelligence. Lecture Notes in Computer Science, vol. 9101. Springer International Publishing (2015), pp.131-140
26. Fiorelli, M., Pazienza, M.T., Stellato, A., Turbati, A.: CODA: Computer-aided ontology development architecture. IBM Journal of Research and Development 58(2/3), 14:1 - 14:12 (March-May 2014)
27. Nair, J., Tudorache, T., Whetzel, T., Noy, N., Musen, M.: The BioPortal Import Plugin for Protégé. In: Bodenreider, O., Martone, M.E., Ruttenberg, A. (eds) Proceedings of the 2nd International Conference on Biomedical Ontology,Buffalo, NY, USA, July 26-30, 2011, p.299 (2011)
28. Pazienza, M.T., Scarpato, N., Stellato, A., Turbati, A.: Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. Semantic Web Journal 3(3), 279-292 (2012)