LinkZoo: A collaborative resource management tool based on Linked Data

Editor(s): Name Surname, University, Country
Solicited review(s): Name Surname, University, Country
Open review(s): Name Surname, University, Country

Marios Meimaris[a,b], Giorgos Alexiou[a,c] and George Papastefanatos[a]
[a]*Institute for the Management of Information Systems, Research Center "Athena", Greece*
[b]*Department of Computer Science and Biomedical Informatics, University of Thessaly, Greece*
[c]*Department of Electrical and Computer Engineering, National Technical University of Athens, Greece*

Abstract. In this paper**,** we present LinkZoo, a web-based, linked data enabled tool that supports collaborative management of information resources. LinkZoo addresses the modern needs of information-intensive collaboration environments to publish, manage and share heterogeneous resources within user-driven contexts. Users create and manage diverse types of resources into common spaces such as files, web documents, people, datasets and calendar events. They can interlink them, annotate them and share them with other users, thus enabling collaborative editing, as well as enrich them with links to external linked data resources. Resources are inherently modeled and published as RDF, and can be explicitly interlinked and dereferenced by external applications. LinkZoo supports creation of dynamic communities that enable web-based collaboration through resource sharing and annotating, exposing objects on the Linked Data Cloud under controlled vocabularies and permissions. We demonstrate the applicability of the tool on a popular collaboration use case scenario for sharing and organizing research resources.

Keywords: Linked Data, Semantic Web, Collaborative Environments, Resource Management, Personal Information Management

1. Introduction

Collaborative management of communal resources is required in many contexts and scenarios, such as collaboration between and within professional organizations, research teams, ad-hoc social groupings that fulfil various purposes and so on. Providing a common representation model for different types of artefacts, such as files, documents, locations, websites, people and events, enables them to be managed and organized under shared principles that ultimately allow for the creation and association of intricate aggregations of different kinds of items with dynamic processes, flows and contexts.

With the re-use of ontologies, codelists and commonly used linked data URIs to annotate artefacts we can provide a layer of semantics that is inherently shared between stakeholders, and helps manage, connect, organize and explore these resources in various meaningful ways. Publication of these resources as linked data enables further exposal for external reference and processing.

Currently, the linked data (LD) initiative focuses on either providing representations of singular entities, or publishing large datasets in flux, created through loosely-defined, context-dependent workflows. In this sense, publication of linked data becomes a complicated, non-universal process, where often custom publishing tools and systems need to be defined in order to deal with domain-dependent complexities. W3C's general guidelines for publishing linked data, as reviewed in the *Best Practices for Publishing Linked Data Note*[1], are mostly concerned with good data modelling (i.e.

[1] http://www.w3.org/TR/ld-bp/

using existing ontologies and vocabularies where possible), following good URI design practices, and providing machine access; in real world applications, most of these issues are primarily solved with ad-hoc design decisions. Except for the publishing needs, most popular semantic tools and systems focus on the storage, interlinking and presentation (e.g., browsing, visualization) aspects of whole linked datasets rather than on the user-oriented manipulation of single information resources. This creates a lack of technical support and functionality when it comes to applying linked data techniques on social and collaborative environments, where data-driven user interaction is intuitive and dynamic. Hence, existing semantic techniques and systems must be complemented by tools that target the operational layer, promoting user-oriented, dynamic interactions between resources at most granular levels.

For example, consider a scenario where a team of researchers get together and start working on a new hypothesis. They start by collecting the literature for reviewing, which can be stored locally in the form of PDF or in remote websites. They annotate and categorize papers based on several properties such as their authors, subjects, venues and journals they were published in and enrich these properties by providing references to external web and linked data resources. Furthermore, they gather relevant dataset sources, experimental data, and information concerning the conference \ journal they intend to submit their work. They share these resources with each other, show part of them to a proof-reader, and collaboratively work and manage their on-going research along all needed resources. In their individual accounts, each team member organizes the same resources differently, by creating distinct folder structures and workspaces that they find more intuitive. Finally, they make available the process or the results of their work as linked data and they enable citation and dereferencing for all resources they managed.

In this paper, we present LinkZoo; a web-based platform for resource sharing and collaboration. LinkZoo is a cloud-based tool that allows users to upload, interlink and organize different types of resources, such as files, websites, people, datasets and calendar events, in collaborative workspaces. It enables sharing and annotation of resources, enrichment with properties and publishing as linked data for citing and consuming. LinkZoo is a Linked Data platform with social characteristics that enables users to form groups and communities by sharing heterogeneous resources, and allows them to collaboratively manage and annotate those using established or custom ontologies. LinkZoo enables users to create views that organize their resources under different perspectives and make them available to others. Finally, it offers an intuitive way of searching over private or public resources and exploring them via a faceted browsing functionality.

We have implemented a web-based front-end, available at *http://www.linkzoo.gr*, that provides a user interface covering all the aforementioned functionality, and we expose the core functions of resource management to developers and applications by making them programmatically available through a REST API as well.

This paper is organized as follows: in section 2 the main features are presented, in section 3 the model and architecture is described, section 4 deals with the system's implementation, section 5 presents the tool's functionality through a detailed use case, section 6 presents related tools and finally section 6 concludes and discusses future directions.

2. LinkZoo Features

In this section we present and discuss the main features of our tool.

2.1. Resource publishing

LinkZoo seamlessly integrates the processes of creating and publishing heterogeneous resources as RDF linked data, and offers a toolkit of common actions for their appropriate management, depending on their type. Items created are published given dereferenceable URIs as well as internal identifiers and are either public or visible to specific users. Moreover, their URIs are citable and can be used by external users for referencing resources without having to link them to their account, similar to the notion of a shareable link in cloud storage systems such as Google Drive[2] and Dropbox[3]. Their representation and metadata can be easily extracted as RDF in a variety of serializations (e.g., XML, JSON, etc.).

[2] http://drive.google.com
[3] http://www.dropbox.com

2.2. LinkZoo resource types and operations

Every item published in LinkZoo is called a resource. In their simplest form, resources are represented using a common model that draws from existing vocabularies for non-functional descriptive properties, such as Dublin Core[4] and FOAF[5]. This accommodates basic metadata assertion, but resources can be further classified into different *resource types* that help specialize the sets of actions that can be performed on them.

LinkZoo handles a variety of resource types (Fig 1), with a common representation layer making it possible for heterogeneous resources to co-exist and be combined to form meaningful aggregations and flows. Current resource types include *files*, *web documents* (i.e., URLs), *people*, *linked data resources* (URIs), *full RDF datasets*, and general purpose *folders*. Folders are special types of resources that organize into contexts sets of resources and can be enriched and linked with knowledge in the same way as other resources.

Sub-types are also possible in order to extend resource actions in more specialized ways. For instance, the *MP3* type extends the *File* type so that instead of merely enabling download, the user can also playback audio, view retrieved music videos and browse similar suggestions based on the MP3 file and its metadata. Extending the list of resource types is a trivial procedure, and can be used to tailor types to the users' specific needs.

Available user actions include creation of a new resource (file upload, URL import or other type-specific actions), creation of new folders, moving, renaming and deleting resources, sharing resources with other users and annotating/enriching resource properties, either manually or in an automatic way.

2.3. Resource sharing and collaboration

Resources can be shared among groups of users in ad-hoc manners. When shared, a resource becomes visible to other users' workspaces and can be manipulated depending on the assigned role(s). A notification mechanism ensures that all involved users are instantly notified about the addition of shared resources in their workspace. Shared resources among a group of users can be viewed and collaboratively edited, with changes being immediately visible to users, as a derivative of the internal model that will be presented in section 3.

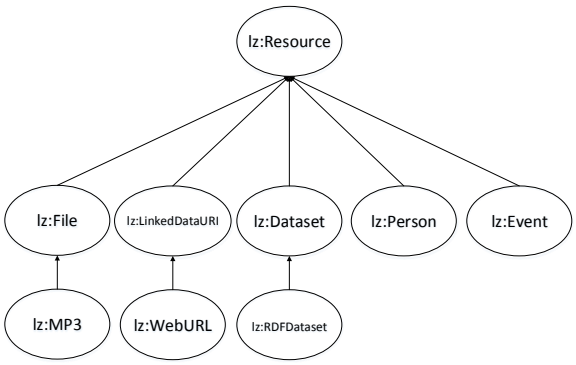

Figure 1. The LinkZoo taxonomy of resource types.

Moreover, each user can organize and explore their resources in their own ways. When sharing folders, their contents are also shared.

The created resources can be processed, annotated, enriched and shared by their users, independently of their type. The roles of *owner*, *editor and viewer* have been implemented. Owners and editors practically have the same rights; the distinction helps to keep track provenance. Furthermore, the notion of *discoverability* is applied to all resources, where resources can be private, i.e. they can only be seen by explicit sharing, or public, i.e. they can be openly discovered and annotated by anyone on the web. Resources can be enriched by assigning new properties to them by any shared editor.

2.4. Annotation and enrichment

Users can assert new facts about resources either manually by annotating them with new properties or automatically by enriching them with knowledge from external knowledge bases. Each resource exists in its creator's account along with all the triples it is a subject of (as will be described in the next section). Most well-known ontologies have been imported in the platform for ease of access via an auto-complete mechanism, but users can use external ontologies and vocabularies by importing them in their LinkZoo account.

Moreover, users are able to define their own properties on the fly under a user-specific namespace and use these properties for enriching their resource with custom attributes. In our example, a researcher wishes to annotate papers under review with

[4] http://dublincore.org/documents/dces/
[5] http://xmlns.com/foaf/spec/

information about their origin, keywords, fields of application and authors, so that she will be able to navigate through them easily based on their properties. Some of these properties can be filled using existing ontologies, which they import. Others, however, may not exist, e.g., a property denoting that a paper is "under review" by a team member. In LinkZoo, she can define the new property, named "underReview", under the user-specific schema and use it at their discretion.

Furthermore, the user can annotate a single resource or a collection of resources, which do not necessarily belong to the same type. Collections of resources can be aggregated in a *drop-zone* and assigned with properties in a bulk manner. Enrichment with properties and interlinking with other resources and external linked data can also be done automatically using external web services. This version of LinkZoo utilizes the Alchemy API[6] in order to automatically enrich incoming URIs based on their content. For example, URL resources are fed to the Alchemy API before importing to LinkZoo. Depending on their structure and content, they can be annotated by Alchemy with relevant DBPedia resources via the generic *rdfs:seeAlso* property.

2.5. Keyword search and exploration

LinkZoo offers intuitive resource exploration by combining *keyword search* functionality over resource descriptions with *property based* filtering. Keyword search is implemented in a close-to-natural-language way, based on the characteristics of a user's available resources. More specifically, keyword search is done in either a one-step or a two-step process. One-step searching is done by entering a keyword and matching it in all possible places of triples, i.e. subjects, predicates and objects. Two-step searching feeds live query suggestions to the user based on the keywords they type. When the user selects one of the suggested queries, the system outputs the relevant results aggregated in a virtual folder. These can then be collaboratively edited, shared, deleted and so on. Text-based search is performed incrementally, by specializing the returned search results with the introduction of new keywords for search and further limiting the result set.

Property filtering is implemented in the form of facets over the list of available properties of the search results. More specifically, when the user navigates to a folder, the system finds all distinct properties of the resources within that folder, and lists them by descending order of the count of their distinct objects. When a property is selected for filtering, its distinct objects will be listed as virtual folders, containing the resources that are linked to them via the selected property. For instance, the researchers can create dynamic facets (in the form of virtual folders) and explore the collected papers by authors, fields of application or any other property their resources are annotated with.

The two methods, keyword search and property filtering, can be combined and applied in an exploratory "find-as-you-go" manner and ongoing results can be stored in multiple views.

2.6. Views

The descriptions of resources enable multiple ways of organization due to the diverse properties and objects found in their triples. The default way of browsing resources is based on an intuitive directory-like manner; still our platform exploits the semantics of a user's resources in order to offer multiple ways of organizing, exploring and searching.

Users have the ability to organize resources, public or private, based on their characteristics and store the results as views. Views do not alter the storage location of a resource; instead they provide different ways of organizing it.

Views can be *static* or *dynamic*. Static views are arbitrary collections of resources where the user is responsible for insertion and removal. They operate very similarly to a virtual folder, where multiple resources are collected and organized explicitly by the user. Static views are different ways to organize the same resources into folders. Users create static views and within them they define different folder structures in order to manage and categorize their resources in various ways. This way, a given resource can exist in two or more views simultaneously.

[6] http://www.alchemyapi.com/

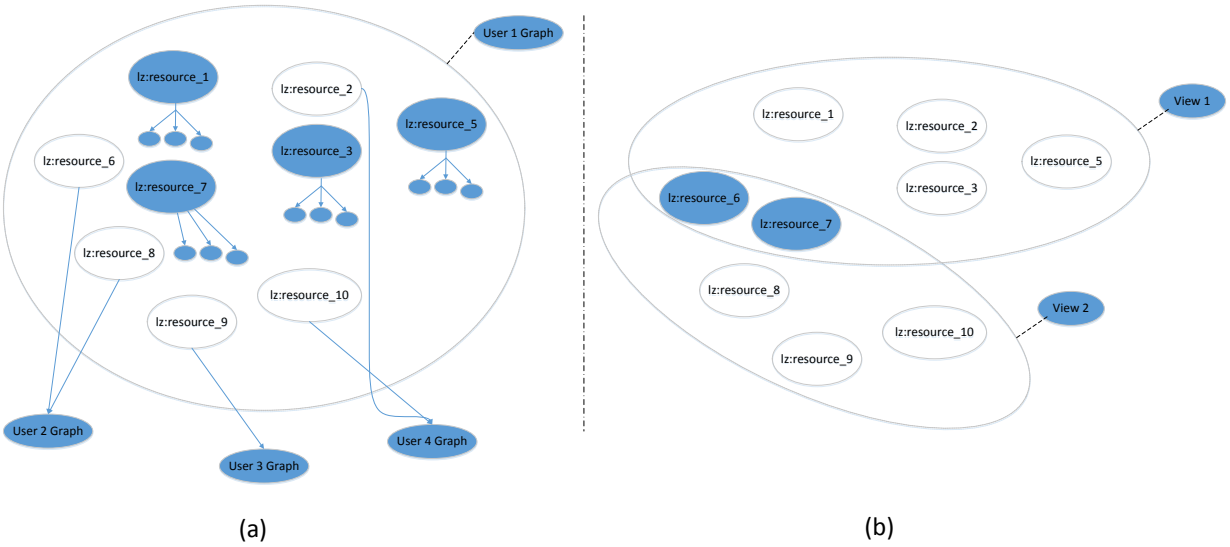

Figure 2 (a)The user graph for User 1. The resources that have their own subgraphs are owned by User 1 (colored with blue), while the resources that point to other external user graphs are shared with user 1 (not colored). (b) The same resources from User 1's user graph are organized into two different views by the user. Resources 6 and 7 coexist in both views.

On the other hand, dynamic views are created based on the results of a search query, thus enabling the user to refresh the contents of the view based on its query definition. For example, one of the users of the research team can ask for "*Files that are under review, are related with dbpedia:SemanticWeb and are shared with me*" and save the result of the query as a dynamic view.

Then, in any given moment this dynamic view can be refreshed to contain the updated result of the query, if for example another member of the team uploads a Semantic Web paper and marks it to be under review. The user has no explicit control over the contents of the container folder that corresponds to the dynamic view. Containing resources are dynamically evaluated based on the query populating the view, which in turn may retrieve resources either from the user graph or other users' graphs. In this way, views leverage semantic web by offering intuitive means to users for organizing, searching and discovering new resources.

3. Model and Architecture

3.1. LinkZoo Data Model

Within LinkZoo resources are represented using a data model that consists of three conceptual parts; the first is concerned with resources along with their functional as well as non-functional metadata, such as resource types, titles, descriptions, comments, identifiers, dates and related users. We have defined a simple taxonomy of resource types that is extensible to accommodate new resource types as well as specializations of existing types.

A top level *lz:Resource* class acts as the common ancestor class for all types of resources. Non-functional metadata such as titles, descriptions, comments, identifiers and so on are modelled by default with the use of commonly used paradigms and schemata, namely RDFS, SKOS, FOAF and Dublin Core. For instance, FOAF is used to model resources that represent people, while Dublin Core properties are used independently of the type. Finally, users can import their own ontologies and recall them on demand, as well as create new custom properties. These are created within a user-specific namespace that is assigned to each user upon registration.

The second part of the model concerns resource administration and privileges over the user base. Specifically, information about ownership, discoverability and sharing is modelled with the use of a set of a small ontology of properties and classes that we have defined. This captures notions such as Private vs. Public for a resource, Owner vs Editor for a user and so on.

The third part contains the definitions of views and the participation of resources in them. Specifically, views provide the way to visually organize resources and can be of two types, namely *static* and *dynamic*, as mentioned in section 2.

The internal data model of LinkZoo makes heavy use of named graphs for defining resources, views, user contexts and so on. In particular, we model the space of each user as a *user graph* that holds information concerning all the resources *owned* by that user. When resources are shared to other users, essentially what is shared is a pointer to the named graph that holds the resource (among other things such as privileges). The user graph contains all triples of resources owned by the user, even the ones contributed by shared users. It serves as a singular access point for its given resources, thus making it possible for other users and applications to gain access to a resource by knowing its URI and user graph, given appropriate privileges and/or discoverability. The user graph is the linked data publishing point that actually contains all RDF triples related to a user's owned and shared resources.

Views are modelled as named graphs as well and they are classified as static or dynamic with the use of appropriate classes of the LinkZoo ontology. A view is represented by a named graph URI and it contains the query parameters in case of a dynamic view, information about a view's base folder as well as folder containment for each resource. This is done with the use of a simple transitive property that denotes folder containment between a resource and a given folder.

Figure 2 displays a user's graph of resources and their organization into views.

3.2. Architecture

The main architectural components of the system are shown in Figure 3. All services that comprise LinkZoo's backend are accessible via a RESTful API, making it possible for different kinds of end-user applications or tools to take advantage of LinkZoo's functionality.

3.2.1. Storage and Data Access Layers

LinkZoo's Storage Layer is built on top of a persistent quad store that supports RDF triple storage, named graphs and SPARQL queries. This is particularly helpful in implementing the LinkZoo model that utilizes named graphs. The implementation specifics of the Storage component have been abstracted in order to make the Storage Layer vendor-independent. The Storage Layer along with the Data Access Layer (DAL) are the two main components involved in data manipulation, storage and retrieval.

3.2.2. Core Functionality Layer

The Data Access Layer communicates directly with the backend's core functionality component, which is comprised of four main sub-components, namely (i) the *Profile Management module*, (ii) the *Resource Action Management module*, (iii) the *View Management module*, and (iv) the *Search and Exploration module*.

The Profile Management module is used for managing the profile data of each user, and is responsible for user administration (creation, deletion of users), user graph assignment and management of a user's associated assets, such as imported and custom ontologies.

The Resource Action Management module implements all actions applied on resources, as well as resource input/output, disk allocation for file upload, and sharing. Different actions are possible for different types of resources, for example files can be downloaded, URLs can be used to direct the browser, MP3 files can be played online and so on.

Furthermore, the Resource Action Management module is responsible for dereferencing URIs of LinkZoo resources and serving content to external applications. This works as follows: the user has the ability to create a dereferenceable URI for his/her resources, much like the "*get shareable link*" function of several commercial cloud storage systems. When this produced URI is dereferenced by a web-browser or a software client, the platform will produce an HTML description of the resource, regardless of whether the client is logged in as a LinkZoo user. Depending on the owner's preference, the dereferenced description of the resource may also give out a link for external users to add the resource to their LinkZoo account, thus effectively becoming co-sharers of the resource.

The View Manager is used for defining and updating static and dynamic views. It is the primary mechanism for defining folder hierarchy and handling metadata about views.

The Search and Exploration Module provides keyword search and property filtering, as discussed in section 2.

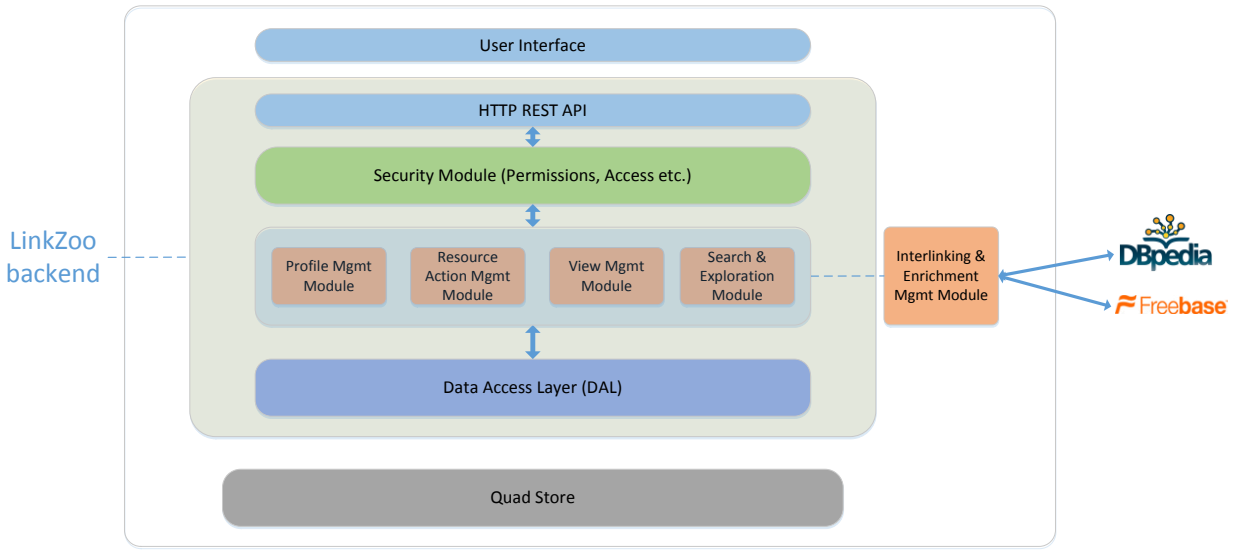

Figure 3 LinkZoo Architecture

It is also responsible for the navigation of the user between folders and/or views, as querying for specific hierarchical relations between resources and folders is easily handled by the search sub-module's query engine. The module allows for on-the-go incremental keyword search over a user's resources, either belonging in her graph or in shared graphs. It is also used to search resources over public graphs. Search is performed on the triples that describe a user's resources, thus semantics play an important role in navigating via keyword search or discovering new items in the public graph.

Finally, the Interlinking and Enrichment module uses external APIs in order to retrieve and enrich resources with facts. Currently, Wikipedia, DBPedia and AlchemyAPI are used for knowledge enrichment, and the FalconsAPI for searching external ontologies.

3.2.3. HTTP API Layer

Incoming HTTP requests are routed in the HTTP API Layer. The requests are evaluated and depending on the parameters, they invoke the appropriate set of controllers after given appropriate access permission by the Security Layer.

3.2.4. Security Layer

Finally, permissions and user privileges are managed separately in their dedicated Security Layer, as most actions require user authentication. User credentials and access tokens are transferred between HTTP requests and evaluated in this layer. Depending on the evaluation result, the requests are passed on or rejected.

4. Implementation

LinkZoo is implemented using open source technologies. Virtuoso 7.1[7] open source edition is used as a quad store in the Storage Layer. The web GUI is deployed on an Apache web server, and the application layer (backend) is implemented with Java 1.7. The application layer is built using the Play Framework[8] for web-based RESTful APIs and deployed on an Apache Tomcat server.

Most of the functionality of the backend of LinkZoo is implemented in a RESTful fashion, in order to isolate business logic from the usage and representation layers. Effectively, LinkZoo is implemented under the Model-View-Controller design pattern. In the following section we describe the technical details of our tool.

4.1. Resource URI minting and handling

Resources are given URIs upon creation based on a mechanism that combines the current timestamp with the user's identifier. Aside from that, unique integer identifiers are assigned to every resource. Upon registration, each user is given a dedicated named graph, which stores the resources created by

[7] https://github.com/openlink/virtuoso-opensource
[8] https://www.playframework.com/

each user, along with their respective triples. In the case of a URL, no additional URI is created, but rather the source URL is kept. Therefore, when two users import the same URL in their accounts, it will have the same URI (its URL) but different identifiers, and will reside in two different user graphs.

4.2. Sharing and Aggregation of Resources

When a resource is shared there is no replication of its description, instead a pointer to the resource's graph is given, thus allowing users to be able to collaborate on the same data object when manipulating a resource, and at the same time inherently keeping provenance information. Hence, the sharing procedure creates two triples on the shared users' graphs, one for their role (e.g. editor, viewer etc.) and one to declare the resource's source graph. Also, sharing of resources generate instant notifications to the involved users, such that they are aware of the resources that are added to their graph.

Table 1 A sample query for retrieving all resources and some basic metadata from the Current View and the Current Folder.

```
SELECT DISTINCT
?resource ?identifier ?label ?type ?mod
 ?role
FROM <userGraph>
FROM <currentView>
WHERE
{
graph ?g{
       ?resource a lz:Resource ;
                 a ?type ;
       lz:identifier ?identifier ;
       rdfs:label ?label ;
       lz:owner ?owner;
    lz:fileModificationDate ?mod ;
       ?role <userURI> .
       FILTER(?type!=lz:Resource)
    }
  ?resource ldfs:rootDir <curDir>
}
```

Incorporation of owned as well as shared resources in the same views, groupings and result sets is enabled by binding the resources' source graphs (i.e. user graphs of their owners) to query variables and performing matching on the resulting set of named graphs. An example of a query that retrieves all resources in a given folder is shown in table 1. Resources originate from the user's graph, as well as all other graphs of users that have shared resources with the specific user. This can be seen in table 1 where the variable *?g* binds to any possible graph URI found in the user's graph.

4.3. Views

Views are also implemented as named graphs containing references to resource URIs. Dynamic views are associated with the search parameters (SPARQL query) of a search operation. These parameters are stored as triples with the view's URI as subject. The user can refresh the view's contents by revaluating the query on the available public or private resources, in effect rerunning the query and refreshing the contents of the view.

4.4. Searching

Searching is implemented using auto-complete suggestions over resource types and property values, in combination with a faceted property filtering mechanism. The resulting query strings are visualized as natural-language phrases (e.g. '*find URLs with rdfs:seeAlso dbpedia:Youtube and linkZoo:owner 'John'*). Search can be limited on a directory, a view or a user's privately shared resources, but can also be done in the publicly shared – anonymous - graph.

4.5. Web UI and API

The web GUI is accessible from web browsers and offers functionality similar to a traditional file explorer. Moving around resources can be done by drag-and-drop. Right clicking on resources brings up a context menu with a list of functionalities such as deleting, moving, renaming, sharing and showing the *properties* pane of a resource.

Through the *properties* pane, users can see and edit the properties associated with the resource, as well as add new triples. New triples can be associated with DBpedia concepts easily via an auto-complete mechanism found in the *properties* pane.

Finally, we have implemented an '*Add to LinkZoo*' bookmarklet that users can add to their toolbars of their browsers, and use it for adding web pages as new resources into their LinkZoo account while browsing. This way, the addition of new web document resources can be seamlessly performed from the browser, without requiring from the user to access his LinkZoo account in another page for this action.

The functionalities of LinkZoo are exposed through a RESTful Application Programming Interface (API), which can be used by developers to create software and tools that connect with user accounts and provide access to their resources. Specifically, all core functions can be called via appropriate HTTP requests. This enables users to search and manage their resources from third party applications such as mobile and desktop apps.

4.6. Authentication and Security

We have implemented a token based authentication for our API. In particular, every time a user logs in, the system generates a random token which is passed and stored on the client-side. The random token is used every time the client-side makes a request to the API and validates the user. This protection mechanism not only assures the user ID but it is also guarding the user account against a particular type of attack, which can occur when a user has not logged out of a web site, and continues to have a valid session. In this circumstance a malicious site may be able to perform actions against the target site, within the context of the logged-in session. This type of protection is known as Cross-Site Request Forgery (CSRF) protection.

5. Use Case

The use case continues and extends our running example (scenario), which refers to a team of researchers that get together to start working on a new hypothesis. So far, the researchers have collected a vast amount of papers for literature; both in PDF format as well as in the form of web documents (URLs), information concerning the conference they intend to submit their work and they have annotated the aforementioned resources with proper semantics (e.g. authors, subjects, keywords, place, time etc.). In order to work on their hypothesis, they have to perform a series of experiments, which involve RDF datasets. Exploiting the LinkZoo's "RDF dataset" resource type, they upload their full RDF datasets, which they can easily share, explore, annotate and generally edit in a simple and intuitive manner. The GUI for the user's workspace is shown in Fig. 4, where resources appear in the middle panel, organized in folders. The current folder path is presented on top of the middle panel, such that the user can easily navigate to folders. On the right side, the property panel lists all available properties of the visible resources and enables property-based filtering. On top, the search bar enables keyword search over the resources. All results are presented in the middle panel. Finally, on the left side (Fig 5) the user has access to the views and the drop zone

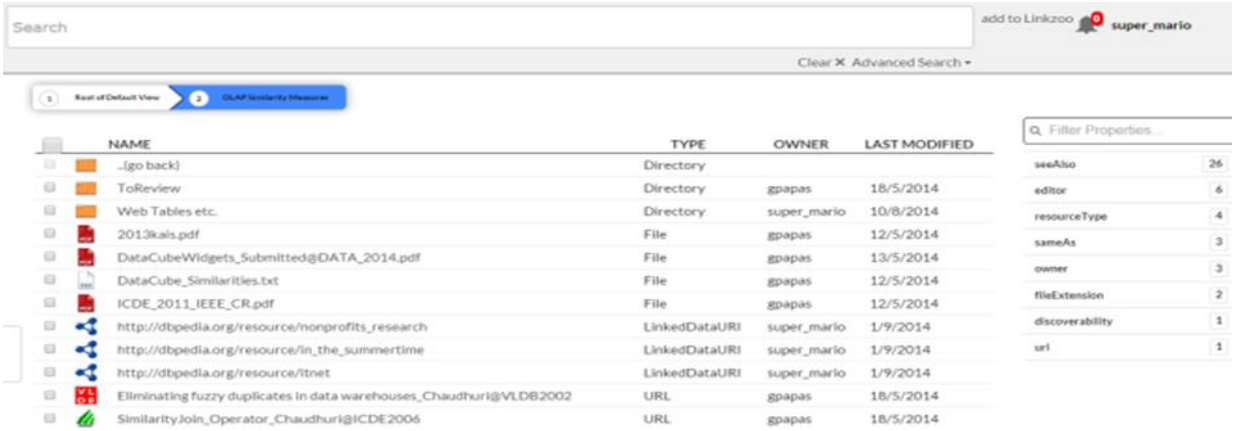

Figure 4 Folder-based exploration, with search on top, and properties filtering on the right sidebar.

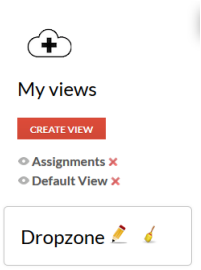

Figure 5 Views and drop zone

Now that the majority of the data (literature and datasets) has been collected, the next step for the team is to make the task assignments for each member. Therefore, they create a "Person" resource type for each member of the team and the project manager assigns each person to one or more resources. For example, the project manager could easily, using the keyword search and exploiting their semantics, categorize the papers for the literature review and assign one or more members of the team to the respective categories / papers. In this way, the project manager as well as the other members of the team could easily track and organize their resources based on the assignments.

For instance, let us suppose that the project manager wants to find quickly what resources from the literature have been assigned to a specific member of his team. There are two ways to do that. The first is to use the keyword search and search for papers that have the property *lz:assignedTo* and the name of the team he wants to track. The second way is to use the property filtering option by selecting resources that are papers. All resources of that type will be listed in the result set and the property list will be refreshed to the available properties of the papers. Out of this list, the project manager will select the property *lz:assignedTo* and all the possible names will appear to the result set.

As mentioned, each member of the team, in his individual account, may organize completely different the same resources that she shares with the other members of the team without affecting the way that the other members organize them. Yet, some members, like the project manager, may want their resources to be organized with multiple ways. In order to achieve that, they may use the LinkZoo's static or dynamic view feature. For instance, from the previous example with the project manager and the assigned members, it is possible that the project manager may wish to have all papers organized not only according to her interests, but also based on their assignments to members. Through LinkZoo she could use the keyword search again (querying about the papers and their assignments) with the "save as view" option, where she is prompt for providing a new name and saving the search results in this new view. In Fig 5 the new view, named "assignments" along with the default one is shown.

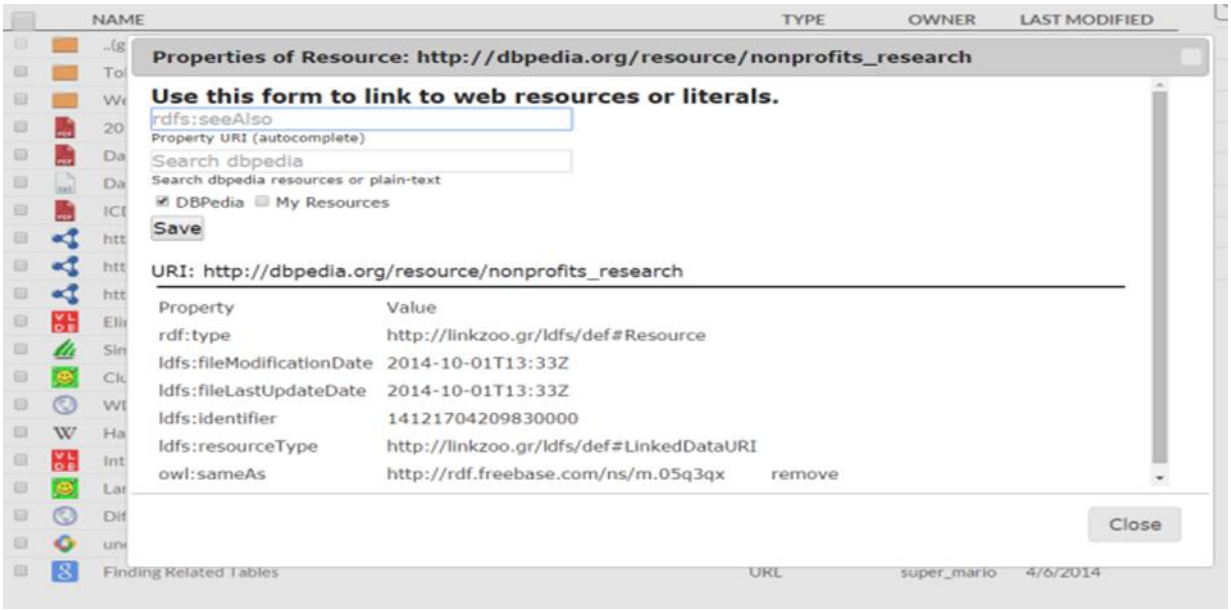

Figure 6 The properties pane for resource dbpedia:nonprofits_research . The top part contains forms that can be used to add more properties to the resource. By default, rdfs:seeAlso is used, but auto-complete draws from imported ontologies to provide suggestions

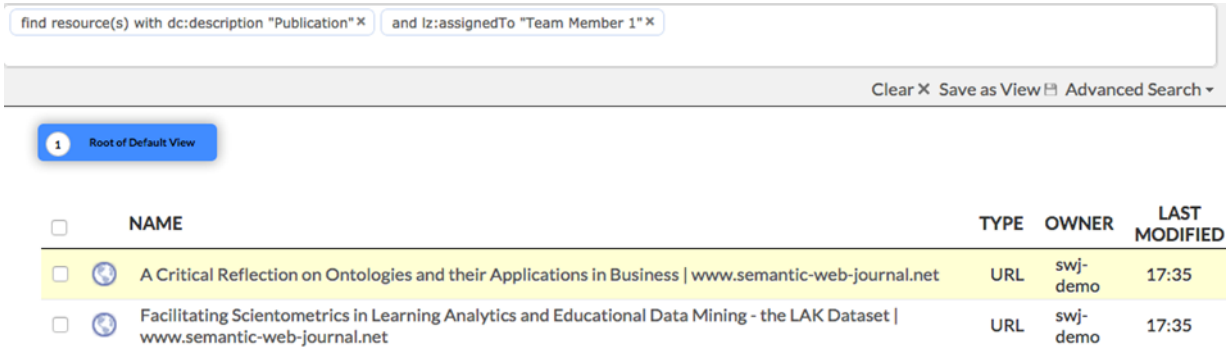

Figure 7 Searching for publications assigned for review to team member 1.

The new view does not affect the way that she has organized her resources in the default way. Moreover, if a new assignment or deletion is made in the resources contained in this view, she can easily track it, just by refreshing the view. This action will automatically update the resources within the view in order to match the query from which the view was created.

Continuing our scenario, two of the members of the team have finished a set of experiments and uploaded the spreadsheet with the results on their private space and annotate them properly. The annotation was made with two external ontologies that the researchers imported in the system. The first researcher has imported, from an RDF file, an ontology about scientific experiments and the second researcher has imported, from a public URI, an ontology about data provenance. These two members are the primary authors of the spreadsheet while a third member does not participate in the experimental process, but is interested in reviewing their work. They can share a subset of their resources with him, e.g. their draft along with a few spreadsheet files with experimental data, by giving him the role of viewer. Moreover, they can annotate this subset with a specific property, e.g., "underReview", which facilitates the reviewer to easily find and organize these shared resources in her own workspace. In case this relevant property is not defined in either of the system ontologies, they can ad-hoc define and in the same step use this new property for annotating these resources. In Fig 6 the properties pane for the resource *dbpedia:nonprofits_research* is shown, where the user can inspect existing or assign new properties. Also, a sample search containing resources marked as publications under review by user 'team member 1' can be seen in Fig 7.

Finally, after they have finished all the process of experiments and writing, the team could easily switch the final version of their paper along with the dataset and the experiments, from private to public and publish them as Linked Data. In this way, they enable citation and dereferencing for these resources. For the interested reader, the aforementioned use case along with a sample workspace is available at *www.linkzoo.gr* under the account *swj-demo\swj-demo* (same as the owner column in Fig 7).

6. Related Tools

In this paper we build upon the work done in [19]. Due to the various functionalities of LinkZoo, we position our work in the intersection of Personal Information Management, Collaborative Platforms, Social Networking, Linked Data and the Semantic Web. To our knowledge there are no approaches that tackle the same collection of issues, however a lot of work has been done on smaller combinations of the aforementioned fields.

Several tools address collaborative semantic editing [2,6,12,13,14,15,16,17,18] however most of these are concerned with ontology editing within closed communities, and thus render themselves as tools for conceptual modelling in communities of experts.

The idea of managing heterogeneous items in the collection of a user's personal information has been addressed by Personal Information Management (PIM) tools, and adding a semantic layer to this gave rise to the Semantic Desktop. In [3,4,5,7,8,9,10,11] systems, tools and frameworks are presented that

apply a semantic layer on user desktops, and combine diverse things such as files and email. In [3] the core aim of the NEPOMUK project was to specify a standard for Semantic Desktop communication and processing. Their work is based on Semantic Web standards and technologies, with RDF serving as a common data representation format. In [4] X-COSIM is presented as a RDF-based semantic desktop that supports information linkage and reuse for email and file management. The Gnowsis Semantic Desktop [5] provides the ability to connect files, contacts and other user assets and assigns URLs to them so that they can be annotated by third party applications. In [1] the authors present TripFS, a system that provides a Linked Data layer on files of existing file systems, by assigning URIs and making them dereferenceable on the web.

Most of these approaches lack the social dimension and do not offer centralized, web-based functionality or space for users to interact and share resources, but rather operate under a more closed-world assumption, with deployment taking place on the client side, and minimum support for external application development. LinkZoo provides a centralized, web-accessible platform that is easy and intuitive to use, as well as accessible from third-party users and applications.

7. Conclusions and Future Work

We have presented LinkZoo, a web-based, Linked Data tool that supports collaborative management of heterogeneous resources between ad-hoc groups of users. LinkZoo is a social platform that leverages the semantic web in order to facilitate collaboration between non-expert users over diverse domains and scenarios.

With LinkZoo, we distinguish between three dimensions of Resource Management tasks, namely (i) acquisition, organization, storage, and retrieval of information, (ii) application types that support the execution of such tasks for specific information items, e.g. file management, text processing, URL resolution and (iii) defining application domains in which application types and task types are combined with respect to a specific domain or context, e.g. biology or collaborative research. Furthermore, enabling social interaction via sharing and collaboration broadens the notion of Personal Information Management to that of Community Information Management. LinkZoo is designed across these dimensions, while inherently it makes use of linked data technologies in order to introduce interoperability and collaboration on the web.

In the future we intend to extend the coverage of resource types and incorporate data from social networks, such as linkedIn, g+ and youtube in order to build thorough social user profiles. We also intend to study scalability and performance issues concerning large user-bases and greedy annotations. Finally, we intend to use the platform as a test-bed for automated integration of information resources on the Data Web.

8. References

1. B. Schandl, N. Popitsch. "Lifting File Systems into the Linked Data Cloud with TripFS." In LDOW. 2010.
2. S. Auer, S. Dietzold, T. Riechert. “OntoWiki - A Tool for Social, Semantic Collaboration”. In ISWC 2006: 736-749
3. A. Bernardi, G.A. Grimnes, T. Groza, S. Scerri. “The NEPOMUK Semantic Desktop”. Context and Semantics for Knowledge Management 2011: 255-273.
4. T. Franz, S. Staab, R. Arndt. "The X-COSIM integration framework for a seamless semantic desktop." In K-CAP2007.
5. L. Sauermann. "The Gnowsis Semantic Desktop for Information Integration." Wissensmanagement. 2005.
6. D. Quan, D. Huynh, D. R. Karger. "Haystack: A platform for authoring end user semantic web applications." In ISWC 2003. 738-753.
7. Groza, Tudor, Siegfried Handschuh, and Knud Moeller. "The NEPOMUK project-on the way to the social semantic desktop." (2007).
8. Sauermann, Leo, Gunnar Aastrand Grimnes, Malte Kiesel, Christiaan Fluit, Heiko Maus, Dominik Heim, Danish Nadeem, Benjamin Horak, and Andreas Dengel. "Semantic desktop 2.0: The gnowsis experience." In *The Semantic Web-ISWC 2006*, pp. 887-900. Springer Berlin Heidelberg, 2006.
9. Franz, Thomas, Steffen Staab, and Richard Arndt. "The X-COSIM integration framework for a seamless semantic desktop." In *Proceedings of the 4th international conference on Knowledge capture*, pp. 143-150. ACM, 2007.
10. Quan, D. A., and R. Karger. "How to make a semantic web browser." In*Proceedings of the 13th international conference on World Wide Web*, pp. 255-265. ACM, 2004.
11. Cai, Yuhan, Xin Luna Dong, Alon Halevy, Jing Michelle Liu, and Jayant Madhavan. "Personal information management with SEMEX." In *Proceedings of the 2005*

ACM SIGMOD international conference on Management of data, pp. 921-923. ACM, 2005.

12. Farquhar, Adam, Richard Fikes, and James Rice. "The ontolingua server: A tool for collaborative ontology construction." *International journal of human-computer studies* 46, no. 6 (1997): 707-727.
13. Palma, Raul, Oscar Corcho, Asunción Gómez-Pérez, and Peter Haase. "A holistic approach to collaborative ontology development based on change management." *Web Semantics: Science, Services and Agents on the World Wide Web* 9, no. 3 (2011): 299-314.
14. Tudorache, Tania, Csongor Nyulas, Natalya F. Noy, and Mark A. Musen. "WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web." *Semantic web* 4, no. 1 (2013): 89-99.
15. Adrian, Weronika T., Antoni Ligęza, Grzegorz J. Nalepa, and Krzysztof Kaczor. "Distributed and Collaborative Knowledge Management Using an Ontology-Based System." In *Artificial Intelligence for Knowledge Management*, pp. 112-130. Springer Berlin Heidelberg, 2014.
16. Holsapple, Clyde W., and Kshiti D. Joshi. "A collaborative approach to ontology design." *Communications of the ACM* 45, no. 2 (2002): 42-47.
17. Noy, Natalya F., Abhita Chugh, William Liu, and Mark A. Musen. "A framework for ontology evolution in collaborative environments." In *The Semantic Web-ISWC 2006*, pp. 544-558. Springer Berlin Heidelberg, 2006.
18. Braun, Simone, Andreas P. Schmidt, Andreas Walter, Gabor Nagypal, and Valentin Zacharias. "Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering." *Ckc* 273 (2007).
19. Meimaris, Marios, George Alexiou, and George Papastefanatos. "LinkZoo: A linked data platform for collaborative management of heterogeneous resources." In Poster & Demo Session, ESWC2014.