Oxford Brookes University
School of Technology

MSc Dissertation

Semantic Mashup to Query Localised Data

Final report

Martin Filliau, `10093736@brookes.ac.uk`

September 1, 2011

Supervisor: Prof. David Duce

Abstract

More and more content is published on the internet and in particular in social media through user-generated content. This growth is a challenge for finding relevant information, and among others, local information. Comprehensive knowledge about a given area can indeed be scattered into disparate web sites. It also seems difficult to people to contribute useful information about their area because of a lack of tools.

This dissertation first explores various ways to retrieve information and to integrate it using domain translation and suitable ontologies. Then, our system is described. It is about the creation of a platform able to consume information coming from agents, reconciliating information from many different data sources about a given area (specifically targeted about Oxford, UK in the prototype). Users can contribute and improve existing information, following principles of a Semantic Wiki. Information collected is about venues and events. It is possible to build powerful queries about local environment, such as finding restaurants opened at a certain time. The prototype has been evaluated by testers (crowdsourcing evaluation) and agents have been evaluated based on metrics. A simplistic augmented reality application has been developed to demonstrate the scalability of the platform.

It appears that building worthy agents can be difficult depending on nature (and quality) of data sources. Integration of various data sources is often problematic due to imprecise data. Tests of the prototype with users show that they find added value and appreciate search features but seem reticent to contribute information.

Acknowledgements

I would like to thank my supervisor, Professor David Duce for his guidance and advices all along this eight months project.

I would also like to thank my friend, Vivien Barousse, for our discussions on architecture and exchanges of points of view on design and code.

Many thanks to my father, Michel Filliau, for his careful proofreading and his support.

Thanks to my circle to bear me since 1988.

And finally, many thanks to anonymous testers of my solution whom, I am sure, will recognise themselves.

Contents

1. Introduction

1.1. Rationale

1.1.1. More and more information available on the Internet

The volume of information available on the Internet explodes, with more and more information published on blogs, micro-blogs and social networks. It is difficult to find information on a restricted geographical area. Information is indeed often scattered into various websites that it may be difficult or time-consuming to find.

1.1.2. A queryable web needs structured data

Getting *comprehensive knowledge* for a given area appears also difficult because information is not structured and hardly findable. For example, getting timetables of opened restaurants may require to visit a large number of websites – without being sure to have a correct answer. This kind of knowledge is often already present on the Internet, but not in a structured way.

1.1.3. The need of a vertical tool to query localised data

There is a lot of tools available on the internet to get information about specific events, more often in a horizontal way (i.e. focusing on a specific subject but at large scale, such as concerts or conferences "everywhere in the world"). But it seems that there is a lack of a tool focused on a vertical way (i.e. focused on a place such as a city), that is to say a *mashup* of existing data sources about a place. It could be used to help people in a defined area, in their everyday life or for helping tourists.

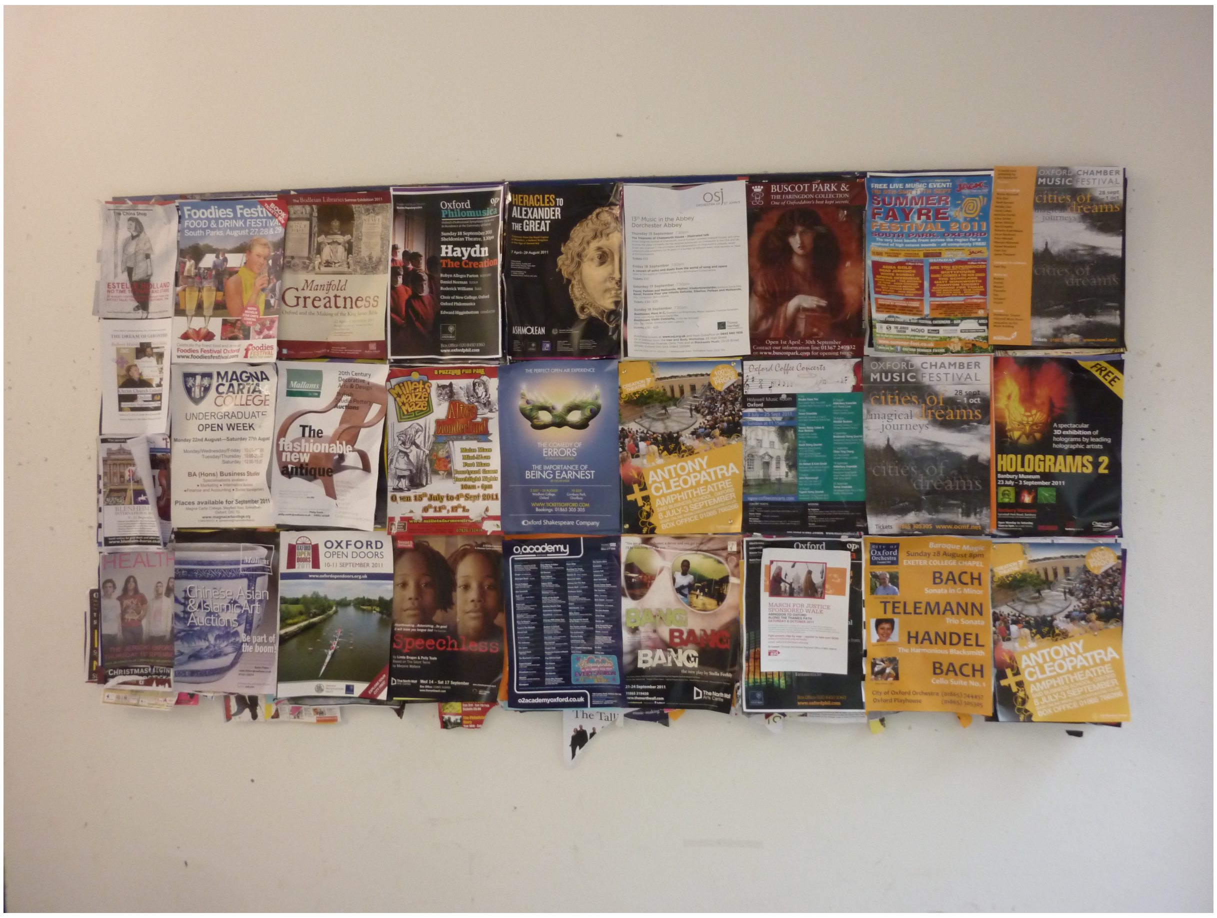


Figure 1.1.: Wall with classified ads in Covered market, Oxford. The aim is to put this "browsing experience" online, in a structured way.

1.2. Outline of the problem

Problem can be broken down in three parts which require different competences.

1.2.1. Information Retrieval and crowdsourcing

First, information should be retrieved from targeted, suitable data sources. This is a challenge because information need to be extracted from web pages (or other formats) to be adapted to our data model.

Another source of information is people who would (happily) share their knowledge. Web 2.0 web applications and wikis often propose users to contribute information into their system (most known example is the encyclopedia Wikipedia).

1.2.2. Structuring information

Information need to be structured: it should fit our chosen data model.

1.2.3. Displaying and querying information

There is two main challenges:

- Data should be well structured to be able to make powerful queries.
- Provide Graphical User Interfaces to:
  - Display information in a simple but effective way.
  - Help user build queries.

1.3. Objectives

The aim of the project is to build a semantic wiki to describe a geographical area (such as a city). Adding a layer of knowledge on top of an area would allow to make complex queries about the local environment.

This goal presumes multiple objectives:

- Define which data to represent, and which ontologies to use to represent data. This includes geospatial and temporal information.
- Find types of potential data sources, integrate and format them.
- Find suitable data storage.
- Build a middleware layer to consume information.
- Build interfaces to:
  - let user contribute structured ("semantic") information.
  - let user make queries

The proposed system should be able to answer to *questions* such as:

- "Where can I find an asian restaurant (i.e. chinese, japanese, vietnamian...) that is open on a Sunday night?"

- “Where is it possible to listen to live music in city centre today?”
- “Is there any shop opened late at one mile around my current position?”

1.3.1. Hypothesis

To be able to answer to such questions, we propose to build the system with the following experiments in mind:

Hypothesis	**Criteria for success**
Users are going to (find reasons to) contribute	Observation, Metrics to calculate users' contribution.
Users contributions are going to *improve* exisiting information.	Observation of the difference between (automated) collected data and their new version.
Users are going to *create* new information.	Observation of facts created by users.
Automatically collected information will be of a sufficient *quantity*.	Observation of new incoming information per day.
Automatically collected information will be of a sufficient *quality*.	Observation on incoming data structure.

Structured information will allow users to make spatio-temporal search and “take the pulse” of a defined geographical area.

2. Research

2.1. Background review

The project is related to various fields including research on:

- Semantic Web
- Middleware and Web Services
- Urban Computing and "augmented tourism"

Thematic of "*Urban Computing*" has some projects such as [Valle, Celino, and Dell'Aglio (2010)] where a system of "semantic pipelines" allows users to ask "what's near me?", data being dynamically fetched from distant web services. [Ceccaroni et al. (2009)] proposes an ubiquitous system where semantic web technologies would be used to describe places of interest by gathering information from sources on the internet.

[Ceccaroni et al. (2009)] also proposes a definition of "*hyperlocal web*" (i.e. "hyper as in linked and local as in location") that fits quite well the aim of the present project. The importance of hyperlocal websites (i.e. displaying information about a well defined area, often a community scale area) is known as significant ([Flouch and Harris (2010)]), and one of the main question about those websites is the problematic of structuring data.

2.1.1. On information retrieval and data integration

Dynamic data integration

Some papers propose a dynamic integration of various (RDF) data sources, such as *Semantic Web Pipes* [Le-Phuoc et al. (2009)] or *Semantic pipelines* [Valle, Celino, and Dell'Aglio (2010)]. It means that data are not replicated locally and everything is fetched on demand (except for caching purpose).

[Langegger, Wöß, and Blöchl (2008)] proposes a "Semantic Web Middleware for Virtual Data Integration on the Web" where a *mediator* component create a "virtual data set" of multiple data sources (a SPARQL wrapper is created for each data source). A client can then create a SPARQL query to this "virtual graph", sources being translated "on-the-fly" (with caching capabilities).

This approach may cause problems when data are not consistent, such as data duplication in results or partially available information (in case of a source out-of-order). But it guarantees data of "last freshness" and could create patterns such as "*data finds data*" [Segaran and Hammerbacher (2009)] where context-information could improve the search of the user.

[Palmonari et al. (2011)] goes further with its "aggregated search of data *and services*" where described services can be discovered dynamically.

Push technologies

Alternatively, some systems are about "pushing" information (a system "A" *subscribe* to queries produced by a system "B"; when a new result is published, system "B" *pushes* new results to the system "A").

The *PubSubHubbub* [*PubSubHubbub project*] project seems to be an innovative approach, and work has been done to have a "semantic web" version with *sparqlPush* [Passant and Mendes (2010)]. This is a relatively new approach but it seems interesting as it doesn't need to crawl a system (a website) at regular intervals – results being directly pushed when they are available.

Data transformation

Some tools such as *Triplify* [Auer et al. (2009)] allows "traditional" web applications running with a relational database to expose RDF resources. [Auer et al. (2009)] also lists initiatives coming from the *W3C RDB2RDF Incubator Group*. Triplify in itself looks interesting but it doesn't have a strong community and updates are rare.

[Nachouki and Quafafou (2011)] propose an approach called "Multi-data source Fusion" to reconciliate various data sources. It details various way (including a literature review) to deal with conflicts at data level.

[Omitola et al. (2010)] explains the challenges to provide an "integrated view of linked datasets" and the necessity to find "join-points" to link these datasets (i.e. reference properties to be able to make links between datasets). [Babitski et al. (2011)] link multiple databases at runtime by semantic annotation of concepts, giving as practical example a "disaster management software".

Approaches from W3C, such as GRDDL (*Gleaning Resource Descriptions from Dialects of Languages*, [*GRDDL*]) or XProc (*an XML Pipeline Language*, [*XProc*]) seem interesting to work on XML documents (that could come from a web service for example). Transformations in RDF seem possible from a lot of dialect, including RDFa and microformats [*Bootstrapping the Semantic Web with GRDDL, Microformats, and RDFa*].

2.1.2. On crowdsourcing and "social" semantic web

[Gruber (2007)] pleads for "*collective knowledge systems*", that is to say making good use of the wisdom of crowds in a structured way. The paper is somewhat disorganized and some of the content is now dated, but it is a good introduction to the main challenges of building a system to "structure collective intelligence".

Getting users to contribute

One of the main problems of the Semantic Web is that it seems complicated to the end-user. As said in [Gruber (2007)], it is difficult to make users contribute structured data as interfaces may not be as simple as a text input where users can contribute free text. Some research has been done to provide better interfaces to manage triplets of information [Davies, Donaher, and Hatfield (2010)], or manage ontologies [Kuhn (2008)]. Main result of their research is an "auto-completion" (or suggestion) service that helps the user to find concepts already existing in the system.

[Matyas et al. (2008)] proposes an interesting approach to encourage users participation. Users are indeed proposed to collect geo-data about their local environment, as a game. It seems to be a very actual and persuasive approach to favorise users contribution. It is going to far for the framework of this dissertation but it is an interesting approach for "going further".

Semantic wikis

[Meilender et al. (2011)] publishes a complete state-of-the-art of semantic wiki engines. It identifies 12 semantic wikis identified as "active" projects (in 2011) and differentiates two approaches in semantic wikis:

- "*ontologies for wikis*": an existing ontology is required to build knowledge upon.
- "*wikis for ontologies*": knowledge is build along with the ontology

[Buffa et al. (2008)] presents *SweetWiki*, now an inactive project and some content is dated, but it still shows a comprehensive list of features of semantic wikis and an interesting architecture (it uses among others GRDDL).

OntoWiki [Tramp, Frischmuth, and Heino (2010)] proposes to build a "RDF-based knowledge base". It seems to be an advanced project, very complete (a demonstration is available, it has support for navigating, visualising, authoring information and some "Linked Data" functionalities [*OntoWiki demonstration*]), with a community and a few years of development (since 2006). They are also doing mobile interfaces in [Ermilov et al. (2011)].

[Torres et al. (2011)] use of the "social force to contribute to the semantic web vision" proposes to realise a wiki at web scale to describe web resources. Methodology used to make the evaluation is described.

[Finelli, O'Brien, and Scannell (2010)] is a project of semantic wiki to integrate as much information (currently mainly cultural information) as possible on the city of Venice (Italy), using Semantic MediaWiki [*Semantic MediaWiki*].

2.1.3. On semantic search, space and time and technological concerns

Semantic search

[Fouad et al. (2010)] on location centric semantic search build a mobile application to integrate different data sources and to display points of interest near the current location of the user, in a quite similar way of [Valle, Celino, and Dell'Aglio (2010)].

[Aart, Wielinga, and Hage (2010)] focuses on cultural heritage. It mixes pervasive computing (with the use of mobile devices to realise the spatial search) and the use of semantic web technologies. Pervasive computing (and mobile devices) are only a practical application of the spatial search, but there is no need of these to perform a spatial search[1]. [Aart, Wielinga, and Hage (2010)] also describes their process for merging and aligning resources about location and their use of SKOS to realise that.

Space and time

LinkedGeoData [*LinkedGeoData*], [Auer, Lehmann, and Hellmann (2009)] proposes to extend the Linked Data principle (linking data sets between them) to geo data. Based on OpenStreetMap, a "free wiki world map", [*OpenStreetMap*], its underlying technology is Triplify (cited previously). This dataset contains a lot of information about venues but they seem very succint. It should be a good start to acquire basic information about venues.

[Janowicz (2010)] explains the role of space and time in semantic web. This is a very important topic that needs to be considered carefully.

Technology

[Ioannou et al. (2010)] describes a system to help with the detection of near duplicate resources using *Locality Sensitive Hashing* (they also provide an interesting background review), but there is a lack of real world implementation of their system.

2.1.4. Visualising localised data

Some papers ([Finelli, O'Brien, and Scannell (2010)], [Aart, Wielinga, and Hage (2010)], [Reynolds et al. (2010)]) are suggesting to use *augmented reality* as a "real-world annotation" tool to help users of their applications to have a better understanding of their local environment. Cited papers all use smartphone applications to display "points of interest" near the user. Layar [*Layar website*] seems to be a simple, popular and easy to develop solution to realise augmented reality applications.

Other papers ([Valle, Celino, and Dell'Aglio (2010)], [Fouad et al. (2010)]) are sug-

[1]Location is acquired independently of the device, there is no need of a *mobile* device to get location information.

gesting a more traditional approach displaying “points of interest” on a browsable map.

2.1.5. Summary

The following table summaries papers on broad topic and include additional references.

Topic	*Major references*	*Minor references*
Data integration	[Le-Phuoc et al. (2009)], [Valle, Celino, and Dell’Aglio (2010)]	Mashups with XQuery and SPARQL: [Ali et al. (2011)], [Langegger, Wöß, and Blöchl (2008)]
Data transformation	Triplify: [Auer et al. (2009)]	[Nachouki and Quafafou (2011)], [Omitola et al. (2010)], [Babitski et al. (2011)]
Collaborative systems and wikis	[Meilender et al. (2011)], [Tramp, Frischmuth, and Heino (2010)]	Venipedia: [Finelli, O’Brien, and Scannell (2010)], [Buffa et al. (2008)], [Torres et al. (2011)]
Geo	[Valle, Celino, and Dell’Aglio (2010)]	Mobile application mashing up data sources using location: [Peng et al. (2010)], [Fouad et al. (2010)]
Participation	[Matyas et al. (2008)], [Gruber (2007)]	[Davies, Donaher, and Hatfield (2010)], [Kuhn (2008)]

2.2. Suitable data sources

We have made the choice to focus on the city of Oxford. Suitable data sources have been chosen because they have a sufficient amount of information available about Oxford (Wikipedia and its "semantic web version", DBPedia [*DBPedia website*] are not going to bring a sufficient volume of data, for example). Volumes of data that each datasource could bring into the system has been evaluated (by the last week of July 2011).

2.2.1. Data sources

GeoNames

GeoNames [*GeoNames*] is a geographical database containing a lot of information about places (mainly suburbs information). It uses about one hundred data sources and people can edit data through a wiki interface. Data are freely available under a Creative Commons Attribution license.

OpenStreetMap & LinkedGeoData

OpenStreetMap [*OpenStreetMap*] is a wiki where people enter (basic) information about places and roads. LinkedGeoData [*LinkedGeoData*] is the "semantic web version" of OpenStreetMap. It provides interlinking with Geonames and DBPedia [*DBPedia website*]. It is not complete and contains very few details about venues, but it stills provide a good basis to begin. Being part of the Open Data movement, data are freely available and under the Creative Commons Attribution-ShareAlike license. LinkedGeoData could bring around 500 venues on Oxford.

Facebook events

A lot of users and companies are publishing events on Facebook. As those events are indexed by Google, and as Facebook publishes microformats for those events, queries in Google can reveal events (limited by a keyword search) and it would be quite easy to retrieve those data. This could be an important source of "fresh" events. It is hard to say how much events it could bring into our system as keyword search is quite limited (e.g. a lot of events are located in Oxford. . . street, London).

Alternatively, the Facebook Graph API [*Facebook Developers Graph API documentation about events* (2011)] could also return similar results.

Data are providen "as-is" without any warranty but freely available (at the moment) [*Facebook's Statement of Rights and Responsibilites*].

Foursquare places

Foursquare Venues project [*Foursquare Venues Project homepage*] is an API realised by Foursquare (startup realising a game on mobile phones where users do some "check-ins" into venues to gain "reputation points"). Their API integrate venues' identifiers from third parties (such as the New York Times), contributing to make links between data sources.

This could be interesting for us to be able to get the maximum possible identifiers for a given venue in order to expose "Linked Data". Foursquare doesn't allow the combination of their data with "*other location databases*", even for research purpose (as a result of a special demand).

Oxford City Council

Oxford City Council's website provides basic information about venues [*Oxford City Council's website local view*]. There is very few value added and no information about the "freshness" of data. License is also not specified.

Visit Oxfordshire

Visit Oxfordshire [*Visit Oxfordshire*] is Oxford Tourist Information agency. They provide a lot of information about events in Oxford. Again, there is no export functionalities and no information about the licensing of content.

DailyInfo

DailyInfo [*DailyInfo*] describes itself as a "guide to Oxford life". It contains a lot of information about local events and venues. Information extraction seems difficult at first sight as there is no export functionnalities and the HTML source code doesn't seem very W3C-compliant. Information is available freely and copy of the content authorised for a “personal non-commercial use”. DailyInfo could bring information about around 500 venues in Oxford and there are in average 30 events per day in their database.

Eventful

Eventful [*Eventful*] is a global directory of events (it seems to aggregate more than 1’200 sources). It seems to have a lot of events and venues about Oxford. It publishes iCal feeds per venue and an API is available. Data are available freely (with a limit rate of calls per day). There are about 100 events per month listed for Oxford.

Songkick

Songkick is a directory of musical events and concerts. An API is available but Terms of Use specify that Songkick data must not be stored in local server. There are about 20 concerts per week listed for Oxford.

University of Oxford - Oxford talks

Oxford talks [*University of Oxford - Oxford talks website*] is a directory of events (mainly conferences) occurring in Oxford. It publishes iCal events, about ten events per week in average.

Lanyrd

Lanyrd [*Lanyrd website*] is a directory of conferences. It contains a few conferences occurring in Oxford. It publishes iCal feed. Very few conferences in Oxford are listed (about 6 per month on average).

OxPoints

OxPoints provides “geolinking information for the University of Oxford” [*OxPoints*] with information about every building, room, library, department...Data are exposed with RESTful web services. Individual resources are available (it could be used for linked data) along with services that operates pre-defined (SPARQL) queries. Resources are exposed in various formats (RDF/XML, JSON…). A SPARQL endpoint is also available. This is an interesting initiative to create an open, reusable data silo containing structured and well-defined information.

Opening Times.co.uk

Opening Times.co.uk [*Opening Times website*] provides a wiki where it is possible to contribute opening hours for every venue in United Kingdom. There are information about 30 venues in Oxford.

2.2.2. Summary

The following table tries to summarize information about data sources that seem to be the more complete to describe our domain (in quantity and quality). We have defined some criteria for quality of data sources:

Accuracy is the source well-known?

Accessibility is it easy to retrieve data?

Consistency are the data consistent?

Freshness is it updated frequently?

Credibility is the source believable and objective?

Legality is it legal to retrieve data?

Source	*Accuracy*	*Accessibility*	*Consistency*	*Freshness*	*Credibility*	*Legality*
OSM/LGD	Yes	Yes (RDF)	Yes	Yes	Yes	Yes
GeoNames	Yes	Yes (RDF)	Yes	Yes	Yes	Yes
Facebook	Yes	Yes	**Uneven**	Yes	**Uneven**	Yes
Foursquare	Yes	Yes (API)	Yes	Yes	Yes	No
City Council	Official	No (HTML)	Yes	**Unknown**	Yes	Unknown
DailyInfo	Yes	No (HTML)	Yes	Yes	Yes	Yes
VisitOx	Official	No (HTML)	Yes	**Unknown**	Yes	Unknown
Eventful	Yes	Yes (API)	Yes	Yes	Yes	Yes
Songkick	Yes	Yes (API)	Yes	Yes	Yes	No
Oxford Talks	Yes	Yes (iCal)	Yes	Yes	Yes	Unknown
Lanyrd	Yes	Yes (iCal)	Yes	Yes	Yes	Yes
OxPoints	Yes	Yes (RDF)	Yes	Yes	Yes	Yes

Data sources need to be carefully chosen. Values in bold in the table shows potential problems with these data sources. Freshness of data is very important in our case. Legality is *less* important as long as it is a private prototype.

Data sources which bring new data (or niche) should be prioritised although it would be important to have the maximum of sources to build a comprehensive view.

There are also some local initiatives such as OxStreets and OxfordCivicSociety that may be sources of events.

2.3. Suitable ontologies

Some parts of our domain of study have already been modelised, and Semantic Web principles shows that it is more than necessary to reuse vocabularies each time it is possible (to favorise interoperability of tools).

2.3.1. Representing geo data

Basic Geo (WGS84 lat/long) Vocabulary [*Basic Geo Vocabulary specification*] is one of the most recognised ontology to describe points of interest.

The GeoNames ontology [*GeoNames ontology specification*] (GeoNames has been described in section 2.2.1) uses the vocabulary of Basic Geo and enhance it with useful properties (such as postal addresses).

Address information can also be represented using the vCard vocabulary [*Representing vCard Objects in RDF* (2010)], as this standard format has been translated as an OWL ontology.

2.3.2. Representing time

OWL-Time [*OWL-Time specification*] is a well-known ontology to describe instants and intervals (of time).

Two ontologies have been found to represent recurring events or time, Temporal [*Temporal Ontology specification*] and DateTime [*DateTime ontology specification*], but they don't seem to be widespread and don't have a community.

2.3.3. Representing events

LODE [*LODE: an ontology for Linking Open Descriptions of Events specification*] is an ontology for linking open descriptions of events. It seems to be more focused on historical events. It uses the OWL-Time ontology for temporal attributes.

The Event Ontology [*The Event Ontology specification*] is a combination of OWL-Time and WGS84 Geo Positioning ontologies (described in 2.3.1). It contains a notion of sub-event (allowing us to be able to describe relationship between events).

2.3.4. Describing companies

GoodRelations [*GoodRelations homepage*] is an ontology for describing e-commerce businesses, but it contains some properties to describe "real-world" business, such as describing opening hours. Google recommends the use of this ontology to describe e-businesses.

2.3.5. Summary

As we have seen, some ontologies may cover more than one domain that we need to define. The following table tries to summarize the different capabilities of each ontology.

"Geo" defines the capabilities of a given ontology to describe (in itself or inherited from another ontology) geospatial information. "Time" to describe temporal information. "Recurrent" defines the ability of an ontology to describe recurrent temporal values. "Business" defines the ability to describe information linked to businesses.

Ontology	*Geo*	*Time*	*Recurrent*	*Business*	*Comment*
Basic Geo	Yes	No	No	No	
GeoNames	Yes	No	No	No	
vCard	Yes	No	No	No	
OWL-Time	No	Yes	No	No	
Temporal	No	Yes	Yes	No	Not widespread
DateTime	No	Yes	Yes	No	Not widespread
LODE	No	Yes	No	No	
Event	Yes	Yes	No	No	
GoodRelations	No	Yes	Yes	Yes	

There is not a "good" choice of *one* ontology but each one has its strenghts and weaknesses. A selection criterion for an ontology would be its community, a decent documentation or if it is largely used over the internet. PingTheSemanticWeb [*PingTheSemanticWeb statistics on namespaces* (2011)] makes statistics about the most used namespaces on internet (this is for information purpose only as methodology to retrieve these statistics is not clear).

2.4. Suitable data storage engine

It is critical to find a data storage engine that is efficient. The following sections will only cover some engines that have been found as more relevant with the following criteria in mind:

- free to use (at least for academic usage)
- good reputation
- strong community support
- available on the Java platform

2.4.1. Data storage engines

Jena

Jena is certainly the most popular Semantic Web framework written in Java. It has a strong community and supports OWL. Two engines are available for RDF storage: TDB (native triple store) or SDB (triple store on top of a relational database).

Sesame

Sesame is a toolkit that contains a data storage engine and a framework to build Semantic Web applications.

4store

4store is a relatively new project built for high performances (scalability over many nodes and datasets of billions of triplets). It provides basic full-text search.

Talis Platform

Talis Platform is a "storage in the cloud" (computing) platform where data are uploaded to a distant service. It provides interesting features on geospatial search, full-text search, but its access is restricted and it is not free nor open source.

SIREn

SIREn is the “Semantic Information Retrieval Engine”. It aims to be more considered as a “web search engine” than a triple-store. It provides Full-Text search (built on Apache Lucene) but no SPARQL queries.

2.4.2. Summary

The following table is a summary of the main criteria of comparison for suitable data storage engine.

Engine	*Jena*	*Sesame*	*4store*	*Talis*	*SiREN*
Reasoning	Yes (OWL)	Yes (RDFS)	No	No	n/a
Geospatial	Extensions (geospatialweb / GeoARQ)	Extension (IndexingSail)	No	Yes	No
Full text	Extension (LARQ)	Extension (LuceneSail)	Basic	Yes	Yes
REST interface	Extensions (Joseki / Fuseki)	Yes	Yes	Yes	Yes
POJO support	Yes (Jenabean)	Yes (Alibaba)	No	No	No
SPARQL support	Yes	Yes	Yes	Yes	**No**
Declared as highly scalable	No	No	Yes	Yes	Yes

[Delbru (2010)] explains differences of approach and potential problems to be encountered with LARQ / LuceneSail compared to SiREN. Main risk in using LARQ / LuceneSail is the slowness of queries due to “costly merge-join between the results of the Lucene index and the results of the triple-store”.

Freie Universität of Berlin is conducting regularly the *Berlin SPARQL Benchmark* [Bizer and Schultz (2011)]. It analyses various triple stores and makes benchmarks on SPARQL queries performances. Regarding data storage engines compared previously, it appears that 4store is much more efficient on high volume of data than Jena TDB (the only two cited in *Berlin SPARQL Benchmark*).

3. Short introduction to the Semantic Web

3.1. Defining the Semantic Web

Tim Berners-Lee first coined the term of "Semantic Web" (in 1999), which he defines as a "web of data that can be processed directly and indirectly by machines". As suggested in figure 3.1, the main idea is to have a *common format* to represent data, data being structured as *resources*, described by *common vocabularies*.

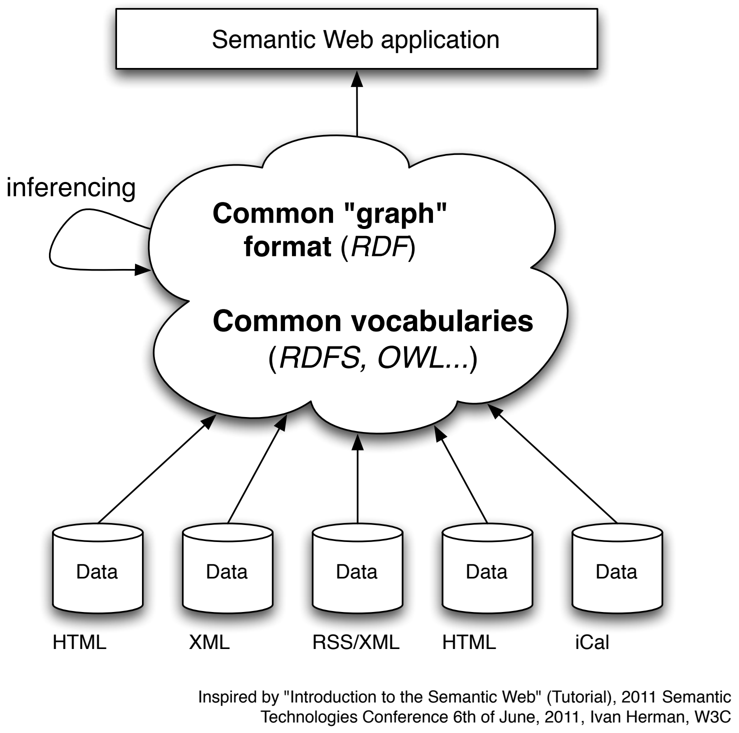

Semantic Web application
inferencing
Common "graph" format (RDF)
Common vocabularies (RDFS, OWL...)
Data
Data
Data
Data
Data
HTML
XML
RSS/XML
HTML
iCal
Inspired by "Introduction to the Semantic Web" (Tutorial), 2011 Semantic Technologies Conference 6th of June, 2011, Ivan Herman, W3C

Figure 3.1.: Big picture of the semantic web

3.1.1. Technologies

W3C (World Wide Web Consortium) proposes standardized technologies for the Semantic Web. The assembly of Semantic Web technologies form a "Semantic Web Layer Cake", shown in figure 3.2. Most commonly used technologies will be described below.

RDF is a standard language to describe *resources* and relations between resources. Information is coded as *triplets* containing:

- *subject* – the identifier of the resource (e.g. "Oxford")
- *predicate* – element of vocabulary to describe the resource (e.g. "contained in")
- *object* – value whether literal value or link to another resource (e.g. "Oxfordshire", which is also a resource)

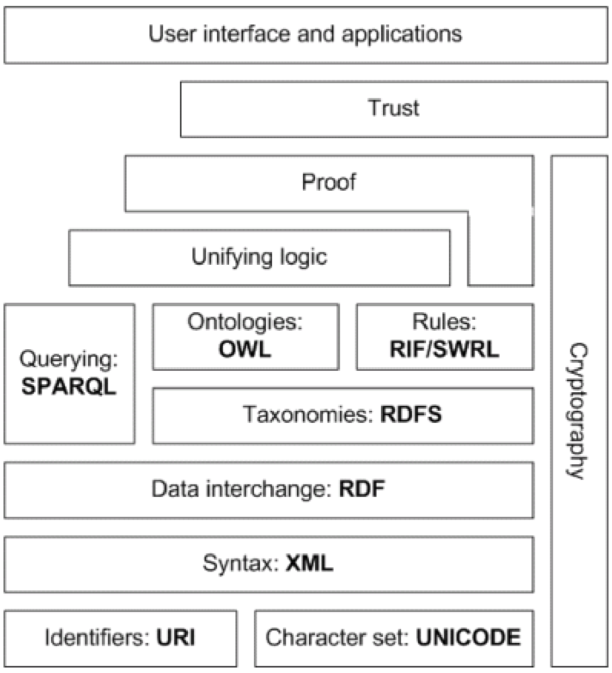


Figure 3.2.: Semantic Web stack

RDF is an abstract language and can be expressed in many *representations* (formats) such as RDF/XML, N3, turtle... RDF data are stored in a data store called a *triple store*.

Information expressed in RDF must use, whenever it is possible, existing ontologies, described themselves in RDF with RDFS or OWL (OWL adds more expressivity to the quite simple RDFS). Thanks to ontologies, new facts can be *inferred* from RDF data. This can be done with an *inference engine* plugged on a *triplet store*.

SPARQL is the main query language of the Semantic Web. It is used to query datasets described in RDF.

More technologies are still in heavy development, such as a standardized language to define *rules* on data (ontologies are sometimes not sufficient, it is not possible for example to infer that a person A is the nephew of the person B because A's father is the brother of the wife of B). Proof and trust rely on cryptography and remain a work in progress.

3.2. Linked Data

Linked data is about exposing datasets to be interlinked by other datasets, with the aim to create a giant graph. A common exchange format – RDF – will provide a basis to allow comprehension between datasets and URIs will be used as global identifiers of those resources. Common properties, such as the OWL "sameAs" (meaning that two resources describe the same "thing"), allow to define that a resource A on a dataset B is an equivalent resource of the resource C on a dataset D.

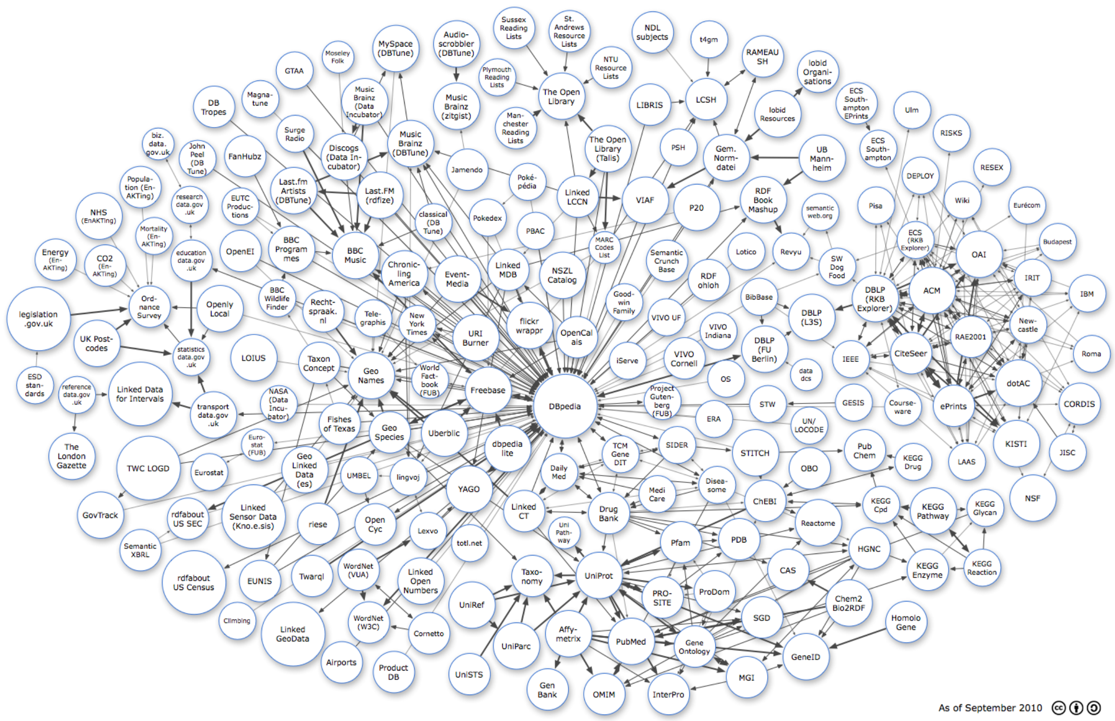


Figure 3.3.: The Linking Open Data cloud diagram as of September 2010. Each circle corresponds to a dataset and each arrow represents a link between two datasets.

A lot of datasets on various topics (biology, movies, encyclopedia...) are exposed and interlinked as shown in figure 3.3. DBPedia [*DBPedia website*], is considered as the “Semantic Web version of Wikipedia” as it analyses the content of Wikipedia (and particularly “infoboxes”) to extract and structure information as RDF.

3.3. Growing interest on Semantic Web and Linked Data

Semantic Web has been a subject of academic research for years but there is now a growing interest in industry too. More and more companies are using semantic web technologies, and Alexandre Passant, CEO of startup Seevl [*Seevl*] gives two reasons to prefer them compared to more “traditionnal ones”:

> “Besides the initial academic exercise, the launch of Seevl.net as a company was motivated by 3 problems that we frequently encounter:
>
> - the need to browse different websites to get a complete picture of an artist / band (bio on one website, genres on another, etc.)
> - the inability to run fine-grained queries (“what’s that band that played with this guy in the 80s”) easily, with a simple UI
>
> . . . ”
>
> [*Seevl An interview with Alexandre Passant*]

These arguments seem appropriate regarding the creation of a “semantic mashup to query localised data”.

4. Methodology

This chapter will cover an overview of the general design process of the prototype.

4.1. Specifications & requirements

4.1.1. Domain

We have chosen to focus on certain aspects of the description of the chosen area (Oxford) to be able to describe them more completly. Prototype will focus on:

- Venues:
  - restaurants, cafes, bars and pubs
  - shops (grocery shops)
- Events:
  - one-time events
  - recurring events (e.g. "live music every tuesday night")

Restricting the domain will ensure a consistent result when testing the prototype. Domain will be easily extensible to new types of venues.

4.1.2. Data model

A simple, abstract data model has been defined following what the domain is expected to describe.

Two main entities are represented, of two different types:

- *Geospatial* objects are described as `Venues`.

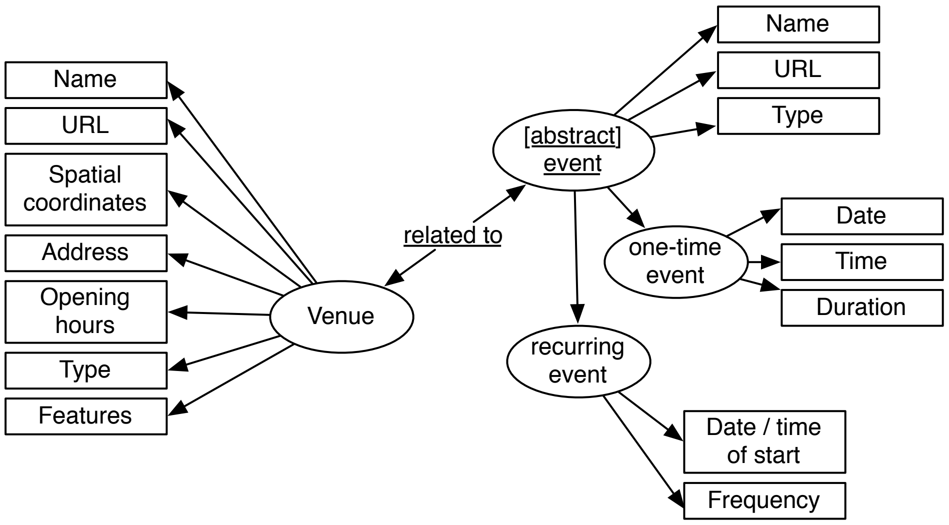


Figure 4.1.: Abstract data model

- *Temporal* objects are described as `Abstract events`.

`One-time events` and `Recurring events` are inheriting from a common abstract structure representing events (`Abstract events`).

Each entity must be identified by an URI in order to be easily recognizable and identifiable.

Temporal and *geospatial* objects are related as an event takes place in a venue. An event could be related to other events (e.g. within the framework of a festival, some events could be part of an "abstract event" that doesn't have any property except of being a container to events).

Description of venues

Geospatial objects should be identified *via* spatial coordinates and an "user-friendly" definition (such as an address). Opening hours of a venue should include general opening hours (per day) and special hours (e.g. opening hours of kitchens for venues where it is possible to eat, happy hours...).

Venues will have one (or more) types:

- *Place to eat or drink*: this type may have sub-types to define types of food.

- *Leisure place*
- *Shop*

Venues have some features:

- *Garden* available
- *Fireplace* available
- *Food served*: if it is a place where it is possible to eat, then type (Take away, restaurant, deli. . .)
- *WiFi* access available
- *Handicap* accessibility

Detecting similarities between venues

When inserting a new venue, the insertion process must check that this venue is not a potential duplicate of an existing venue. Detecting similarities (i.e. record linkage) between venues should be made on some factors:

- Same *address / coordinates*: if spatial coordinates are close, then it might be a duplicate.
- *Name*: distance between names of venues should be compared, to find when some names are approaching and might be the same venue (with some algorithms such as Damerau-Levenshtein).
- Same *type* (if provided)

Possible duplicates should be flagged based on these similarity measures.

Description of events

Temporal objects should be identified by a name. Date and time (or frequency) should be clearly identified. Recurrence should be defined by a frequency (e.g. daily, weekly, monthly), with a date of start and (eventually) a date of end of the recurrence.

Types of events include:

- *Quiz*

- *Live sport*
- *Live music* / concerts
- *Conference*

If a one-time event created corresponds to a recurring event, the link must be established. Creation of a one-time event can be done if there is a particular description to add to this event, or particular hours for example. From a user point of view, it corresponds to the same event.

Detecting similarities between events

On the same principle than venues described in 4.1.2, potential duplicates events should be flagged on factors such as:

- Same *venue*
- Same *date / time* (rounded)
- Same *type*
- *Name*: distance between the name of events might be a clue.

4.2. Overview of the design

4.2.1. Core server

An application running on a server will be in charge of the management of data and its distribution to users.

4.2.2. Agents

Agents are independent softwares that interact (in both way i.e. query and inform) with the server. Their main job is to “push” information retrieved from data sources to the server. They can run from any computer as long as they have access to the server using an HTTP connection.

4.3. Chosen technologies

Development should be done using programming languages of the Java platform, known by the author, to be able to focus on “Semantic web study”.

4.4. Risks & ethical issues

Following risks and ethical issues have been identified as critical for the success of the project.

4.4.1. Domain of analysis too big

For a given area, a lot of data sets and various data sources may be available. If the domain of analysis is too big, then there is a risk of being overwhelmed by disparate data without having a consistent domain and results.

4.4.2. Lack of available data

Alternatively, a lack of available data to transform for our system may make the testing of the system difficult.

4.4.3. Data store problems

Performance problems (high rate of errors on queries, slowness. . .) of the chosen data store may also make the testing of the system difficult. Data store must be chosen carefully.

4.4.4. Difficulty of performing trials with real users

Performing trials with real users may be difficult, mainly to get interesting results. In order to minimize this risk, testers should be chosen carefully as trustable. A short introduction should be realised and questionnaire should not be too long to keep testers concentrated and not bored.

4.4.5. Ethic of information retrieval

Retrieving information from sources (such as parsing HTML pages or RSS/XML feeds, exploiting microformats...) may not be legal. Source must be checked carefully and, if needed, written permission should be asked.

4.5. Project plan

Following chapters will cover the development of the solution. The solution has to be realised in this order:

1. Create an ontology able to describe our domain (chapter 5)
2. Develop the core server (chapter 6)
3. Develop agents that will interact with the server (chapter 7)
4. Develop an augmented-reality application to demonstrate the abilities of the platform (chapter 8)

GANTT diagram showing time consumed for each task is available as appendix F.

5. Building an ontology to support the prototype

Following abstract data model defined in section 4.1.2, an ontology must be written. Ontology will be described with OWL-DL (Description Logic), it gives the maximum expressivity and guarantees to be processed by any triplet store. The application Protege [*Protege website*] will be used to create the ontology and define concepts.

5.1. Choosing ontologies to build upon

According to principles of the Semantic Web, existing vocabularies must be reused if possible. Following research done in section 2.3, vocabularies to describe geo points, address information, and opening hours will be reused from existing ontologies.

Geo points will be described using the vocabulary of Basic Geo (WGS84 lat/long) [*Basic Geo Vocabulary specification*]. It is the most widespread ontology and it is very simple to express latitude and longitude. Address information (street, postal code, city) will be described using the vCard vocabulary [*Representing vCard Objects in RDF* (2010)] because it is a well-known and widespread vocabulary.

Opening hours of businesses will be described using the GoodRelations ontology [*GoodRelations homepage*] because it is well defined and documented.

5.2. Defining classes and properties

Each resource will have the standard RDFS label property (supposed to be a property readable by humans that describes the resource) filled as the main name (of a venue) or title (for an event).

It has been decided not to use any existing ontology to describe the following domains:

- *event*, because found ontologies don't provide a neat solution and would have been to heavy for the ontology written for the framework of this dissertation
- *recurring event* properties, no simple solution has been found.
- *businesses* will only be described thanks to a RDFS label with no further details (apart the qualification of the RDF type property).

5.2.1. Classes

Our ontology is composed of four main classes which describe main concepts of our domain:

- "Venue", which contains a hierarchy of classes representing the different types of shops, leisure places and places where it is possible to eat[1] (it contains a tree representing different types of food such as "French", "Italian", "Greek"...)
- "Event", which contains different classes representing types of event (e.g. concert, conference...)
- "Day" represents a day of the week (it will be used to define recurrences)
- "EatPlaceType" represents a type of place where it is possible to eat

5.2.2. Entities

Entities in an ontology are a way to describe "static" resources. It is used in our ontology for:

- the class "Day", it has many *entities* representing each day of the week (Monday, Tuesday...)
- the class "EatPlaceType", it has entities representing types of place to eat, such as "Café", "Restaurant", "Pub", "Take away", "Home delivery". A "place to

[1]An alternative design decision could be, instead of having an OWL hierarchy of classes, to represent types of food as SKOS concepts. Having a hierarchy of classes, we directly use defined ontology to infer that, for example, "Chinese food is asian food". But it may be harder to maintain and to extend, as it is necessary to modifiy the ontology each time a new type food has to be added.

eat" can indeed have more than one "EatPlaceType" as it is possible that a restaurant also do home delivery, for example.

5.2.3. Properties

The ontology is composed of three *object* properties (i.e. link between resources):

- "hasEatPlaceType" is a relation between a "place where it is possible to eat" and a "type of place where it is possible to eat".
- "hasVenue" is a relation between an "Event" and the "Venue" where it occurs.
- "occursOn" is a relation between a (recurring) "Event" and a "Day" (when it occurs).

Ontology is composed of ten *data* properties (i.e. literal values):

- "hasIdentifier", functional property (i.e. value of the property must be unique in the triple store) to give an identifier (as a string) to each venue and event
- "beginAt" describes the date of the beginning of an event
- "endAt" describes the date of the end of an event
- "fireAvailable" indicates if a fireplace is available
- "gardenAvailable" indicates if a garden is available
- "hasGlutenFreeAvailable" indicates if gluten-free menu is available
- "hasHandicapAccess' indicates a wheelchair-friendly venue
- "hasPrice" indicates the price of an event[2]
- "hasUsualHour" indicates the time of a recurring event
- "wifiAccessAvailable" indicates if a Wi-Fi access is available

Full ontology described as OWL/XML is available as appendix B and a visualisation of classes hierarchy is available as appendix C.

[2]If this property is absent, it means that the price is *unknown*. It this property is present and the price equals zero, it means that the event is free. It is quite simplistic as it doesn't allow an event to have different types of prices (students prices, members prices...).

5.3. Preparing the integration of the ontology to the application

In order to be able to easily manipulate our data, used ontologies(i.e. classes, properties and individuals) will be translated as Java classes containing properties of ontologies[3].

This is done automatically thanks to the tool *schemagen* of the Jena framework. When a modification to an ontology is done, building the module automatically generates new Java classes ready to use by every application (i.e. server, agents. . .) that would need to use an ontology. It allows us to be sure that existing properties are used, otherwise it will not be possible to build (and compile) the application.

[3]It is different from binding Java objects to RDF, as an Object-Oriented-to-RDF mapper would do. Our approach follows more the RDF model than trying to reproduce Object-oriented programming patterns.

6. Core server design

The core server will handle communications with agents and users of the application. It has been developed with Java EE technologies, and in particular EJB (Enterprise JavaBeans) 3.1.

Figure 6.1 is an overview of the architecture and it does not include practical features (such as logging). Also, it has been chosen to not provide any security feature (such as user authentication, access control…) to only focus on the core functionalities of the prototype.

Arrows represent direct connection between components (for example, no component except the “Data Access Service” can access the Semantic Data Store).

Rectangles named “agent” or “user” represent independent software (in the case of agents) or humans that will interact with our core system. Humans will be able to use two interfaces: a simple – interactive – web interface and a query interface to execute SPARQL queries. Agents will use RESTful web services.

Rectangles in middleware and presentation layers represent independent services and will be described below.

This multi-layers architecture is made to loose-couple components and be able to change some components more easily (it should be easy to change some parts of the application, such as moving to a different data storage without having to modify the integrality of the application).

6.1. Data storage

It represents the database where *our* data (consolidated view of information aggregated from many sources and user input) will be stored.

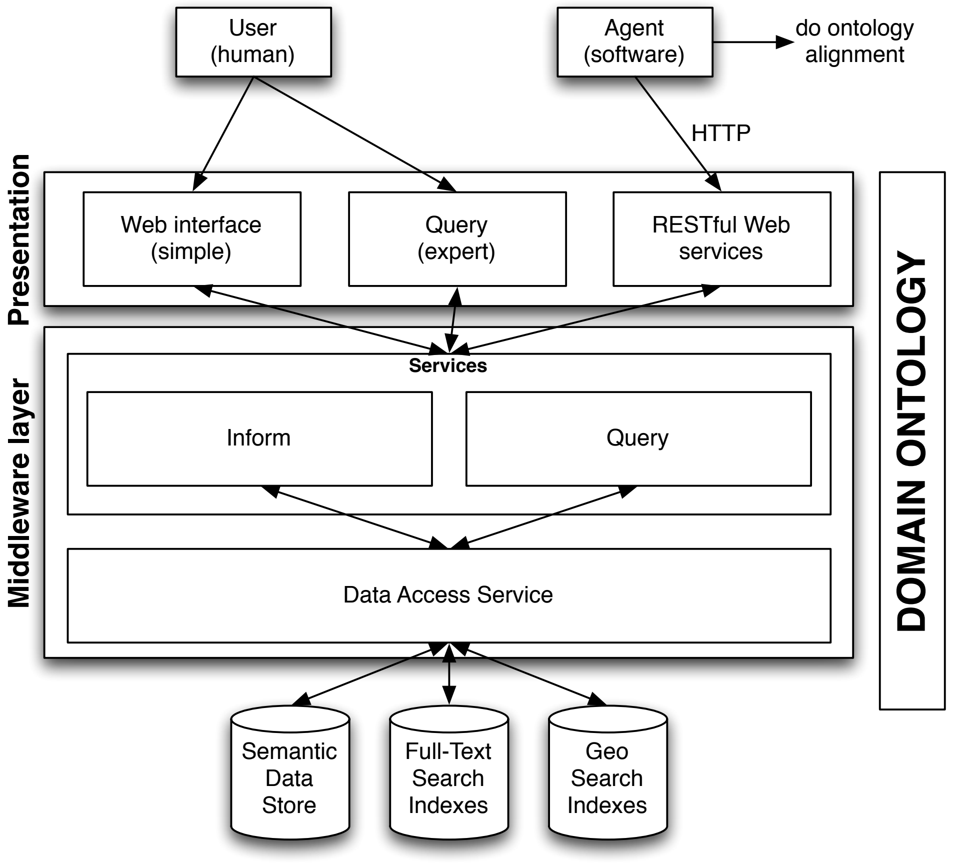


Figure 6.1.: Overview of the architecture

The datastore must be chosen carefully. Criteria for finding a suitable datastore should be:

- Triple store (ability to execute SPARQL queries)
- Reasoning capabilities (or API to plug a reasoner).
- Have full-text search capabilities.
- Support geospatial queries.
- Documentation available and strong community support.

Scalability is not a criterion of utmost importance as the volume of data expected should not exceed a million of triples.

Following research done in section 2.4, it has been decided to use Jena toolkit and its triple store called "TDB". As it does not include full-text search capabilities or support of geospatial queries[1], extensions LARQ [*LARQ website*] for full-text search and GeoARQ [*GeoARQ website*] for geospatial queries will be used.

Jena "TDB" is stored as files on the file system, LARQ and GeoARQ are Lucene indexes, also stored as files on the file system.

6.2. Middleware

Middleware layer is the core of our system. It contains all the *business* logic, organised as *services*.

6.2.1. Data Access Service

It is a service which handles and encapsulates every connection made to the semantic data store. This should ensure that the database and the application are loosely coupled. This service is an EJB Singleton which manages connections to the data storage and (Lucene) indexes.

When the application starts, it instantiates the data storage and load ontologies (provided by the Java library described in section 5.3).

This service provides two main functions: persist RDF statements in the triple store and query (SPARQL) the triple store.

Persisting RDF statements

When RDF statements are "sent" to this service, they are first indexed in two ways by the two extensions cited in section 6.1:

- LARQ (full-text index) indexes all statements containing *literals* (i.e. string).
- GeoARQ (geo index) indexes all statements containing latitude or longitude expressed with the Basic Geo (WGS84) vocabulary (described in chapter 5).

RDF statements are then added to the model (i.e. the triple store).

[1]It would be possible to make geospatial queries using SPARQL to query on a bounding box, but it seems slow and limited (for instance, it would not be easy to sort results).

Querying the triple store

Data Access Service exposes a method to make SPARQL queries. Thanks to SPARQL extensions, both triple store and data contained in Lucene indexes are queried in an uniformized way.

6.2.2. Services

Services are higher-level components in charge of global operations. They encapsulate every access that is done to the Data Access Service described in section 6.2.1.

Query services

Those services encapsulate in Java classes most commonly used SPARQL queries executed in upper-level layers.

Parameters of queries are optionnal (passing a null value to the function will ignore the part of the query responsible of this parameter). SPARQL queries that can be executed by calling Java methods do the following:

- Find venues of a given type, near a given latitude / longitude, opened at a given day and time
- Find events of a given type, near a given latitude / longitude, occurring at a given date and time
- Find recurring events of a given type, near a given latitude / longitude, occurring at a given date and time
- Find the URI in *our* system of a venue (or event) corresponding to an URI of another data set, called the co-reference.[2].

Results of generated SPARQL queries are the URI of the objects found.

URIs of resources are then used to find all information about those resources and Java objects corresponding to those resources are returned by each method.

[2]For instance, the venue with URI `http://www.dailyinfo.co.uk/reviews/venue/826/Cafe_Tarifa` corresponds in our system to `http://ol.filliau.com/resources/venue/rnCBd`. It is useful for agents to be able to quickly find the equivalent URI.

Query services cover most of the needs for presentation layer, but it is still possible to create your own query.

Examples of SPARQL queries

Services are building SPARQL queries that are sent to the Data Access Service (described in section 6.2.1). Listing 6.1 shows a query whose result is URIs of all shops known to be open on Monday at 14:12, at 0,5 mile around the geo position (51.7527210, -1.2524490). It demonstrates the mix of vocabularies, using ontology build for the use of this application (described in chapter 5), the GoodRelations ontology and the GeoARQ function (line 9), which is a shortcut to query a Lucene index.

Listing 6.1: Query to find URIs of shops opened on Monday at 14:12

```
1  PREFIX ol: <http://oxford-live.co.uk/ontology#>
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  PREFIX gr: <http://purl.org/goodrelations/v1#>
4  PREFIX geoarq: <http://openjena.org/GeoARQ/property#>
5  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
6  SELECT ?r
7  WHERE {
8   ?r rdf:type ol:Shop .
9   ?r geoarq:nearby (51.7527210 -1.2524490 0.5) .
10  ?r gr:hasOpeningHoursSpecification ?o .
11  ?o gr:hasOpeningHoursDayOfWeek gr:Monday .
12  ?o gr:opens ?open .
13  ?o gr:closes ?close .
14  FILTER(?open <= "14:12:00"^^xsd:time) .
15  FILTER(?close >= "14:12:00"^^xsd:time) . }
```

Listing 6.2 shows how to get URIs of concerts occurring on August 15th, 2011 all day, whose venue is 0,1 mile around the geo position (51.7510,-1.2410).

Listing 6.2: Query to find URIs of concerts occurring on August 15th

```
PREFIX ol: <http://oxford-live.co.uk/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX geoarq: <http://openjena.org/GeoARQ/property#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?event
WHERE {
 ?event rdf:type ol:Concert .
?event ol:hasVenue ?venue .
?venue geoarq:nearby (51.7510 -1.2410 0.1) .
?event ol:beginAt ?time .
FILTER(?time >= "2011-08-15T00:00:00Z"^^xsd:dateTime) .
FILTER(?time <= "2011-08-15T23:59:00Z"^^xsd:dateTime) . }
```

Listing 6.2 shows how to get URIs of concerts occurring every Monday, whose venue is 0,1 mile around the geo position (51.7510,-1.2410).

Listing 6.3: Query to find URIs concerts occurring on Mondays

```
PREFIX ol: <http://oxford-live.co.uk/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX geoarq: <http://openjena.org/GeoARQ/property#>
SELECT ?event
WHERE {
 ?event rdf:type ol:Concert .
 ?event ol:hasVenue ?venue .
 ?event ol:occursOn ol:Monday .
 ?venue geoarq:nearby (51.7510 -1.2410 0.1) . }
```

Inform service

This service manages the process of inserting information into the database. Using the Data Access Service (described in 6.2.1), it persists information in the triple-store after testing that incoming data are correct. Each type (i.e. event and venue) has two methods to create and update data.

In any case, it is imperative to know where the data come from. The origin of the data must always be attached to the data itself. This is required for security and legal reasons. It is indeed possible to have to delete some data based on their origin (in case data source becomes inconsistent or restricted access to the source, for example).

Inserting a new venue or event Some checks must be done first, in order to ensure that entered data are valid. Even if the datastore is ontology-enabled, some checks such as checking that the persisted venue has its geo-coordinates in Oxford remains the responsibility of the instance of the application (and its configuration file), as the scope of the ontology is not specific to Oxford. In the case of an event, it is mandatory to check that it is correctly associated with a venue.

Creation of an item also means generating a unique URI. As the name of a venue (or event) may change over time, it is best to produce a unique key as a short chain of characters (potentially memorisable too). URI in our system look like `http://ol.filliau.com/resources/venue/rnCBd`, where "venue" shows the type of resource and "rnCBd" is the unique identifier of the resource.

Updating an event or venue Update process is quite similar to the insertion of a new venue. It also checks that the venue or event already exists in the data store before realising the update.

6.3. Utility services

Utility services are available as helpers of main services.

Geocoding service aims to "translate" human-readable postal addresses as geo coordinates used by our system. It encapsulates connection to the Google Maps geocoding service. This service has been chosen because it is free and very powerful (allowing queries containing fuzzy address, postal code or even building names).

String service helps with translating Java date objects as string used by SPARQL queries (e.g. from a Java date object, get the name of the day). A method is also provided to help cleaning a search string with a list of stop words (such as "café", "bar", "the"...) that were making some search uneasy.

Configuration service encapsulates values specific to an instance of the application such as:

- bounding box of the area where venues will be (upper and lower latitude and longitude)
- path of data storage on file system

6.4. Presentation layer

Presentation layer is composed of a web application and RESTful web services. Both are using services (described in section 6.2.2) as a common "glue" to access data.

6.4.1. Web application

A graphical user interface has been realised. HTML web pages allow users to make the following tasks:

- Search for venues by place, opening hours and type
- Search for events by place, time and type (concerts, conferences. . .)
- Contribute opening hours to a venue
- Contribute a new event to a venue
- Contribute a *recurring* event to a venue

Figure D.1 shows a form to search for a place to eat. Main criteria of search are the type of food, type of "delivery" (e.g. "home delivery", "restaurant", "pub". . .) and the coordinates of venues to find (radius around an address or postcode). It is additionally possible to filter search results with venues known to be open at a given day and time, or with known gluten-free menu, wifi or disability access available. Values in menu lists (food types, and types of venues) are loaded from the ontology.

Results of a search are displayed on a map with bubbles corresponding to results. Alternatively, a list of results is displayed under the map, as shown in figure D.2. From these views, it is possible to click on links on venues and events to have more details about the venue and its events.

Users of the web application can also contribute information such as opening hours of a venue as shown in figure D.3.

It is also possible to add an event to a venue (figure D.4) with a simple form.

Alternatively, if the user is aware of an event occurring recurrently in a venue, there is a special form to fill in, as shown in figure D.5.

The web application has been realised with JSF (Java Server Faces) 2 and the graphical components of Primefaces. It is seamlessly integrated with EJBs (common services). Maps are displayed thanks to Google Maps.

6.4.2. RESTful web services

The aim of RESTful web services is to provide an access on data contained in the application. By using mainly HTTP and XML, key strength of RESTful web services is interoperability with other systems (allowing other applications to re-use our data and keeping the door open to a choice of different technologies for agents).

Jersey [*Jersey website*] toolkit is the main toolkit to publish RESTful web services for Java EE. It is fully integrated with EJBs (services of the middleware layer described in section 6.2.2).

Following sections will describe two ways of accessing data:

- access data thanks to linked data resources
- query data thanks to a SPARQL endpoint

Resources exposed

RESTful web services act as shortcuts to SPARQL queries with most common queries ready to use. It is easier for agents than doing queries (although it is possible). RESTful web services are using services described in section 6.2.2.

Most of the resources are exposed with various *representations*, representation being chosen when the HTTP query is realised via HTTP headers:

- RDF serializations (RDF/XML and N3), for the use of agents
- HTML, for human users

Event and Venue It is possible to get (with the HTTP GET method) an event or a venue by its ID. Following principles of linked data (as described in section 3.2) those resources are exposed and can be processed and interlinked with other data sets.

Each resource exposes the list of each sources for a given venue or event, so it is possible for an agent to automatically follow the links to other sources that represent the same event / venue. It means that *our* data set is linked with other datasets having being parsed by agents (*but* some datasets may not share information as RDF[3]).

[3]It is at least identified that different resources are describing the same "thing" but it may be the only information available as most of the sources only contain HTML.

Additionally, these resources are used to insert and update event using respectively the HTTP PUT and POST methods (as RDF/XML or N3).

Events and Venues It is possible to search for events and venues corresponding to an URI, for a given date / time, for a given location. The HTTP GET method is used.

Geocode This resource helps agents[4] to find the geo coordinates (latitude and longitude) of a point. Giving as parameter an address, this service returns latitude and longitude. It currently acts as a proxy to Google Maps web services.

Search This resource is mainly used for developers to realise some tests with indexes (it is possible to query them directly without having to make a SPARQL query).

Status This resource is not a resource used to expose data to the world, it is used as a way to interact with the application and launch administrative tasks (such as launching an indexation of resources or checking the status of indexes).

SPARQL endpoint

A SPARQL endpoint is available. Any application or user can make queries to the system by emetting an HTTP GET or POST request to a specific URL. Result is given as SPARQL-XML or JSON.

Example of a query

As a practical introduction, let's assume that we want an RDF/XML version of the resource `http://ol.filliau.com/resources/venue/vOYRY`. In the following example, `http://ol.filliau.com/` corresponds to the address of the server hosting the application and `http://oxford-live.co.uk/ontology` corresponds to the namespace of the ontology.

Following HTTP query is realised (some headers have been stripped for readability purpose) specifically asking for an RDF/XML version (line 3):

[4]It is not mandatory for agents to use this service to geocode their data.

Listing 6.4: Query to find URIs concerts occurring on Mondays

```
GET /resources/venue/vOYRY HTTP/1.1
Host: ol.filliau.com
Accept: application/rdf+xml
```

Response headers confirm that the content type is RDF/XML (line 4):

Listing 6.5: Query to find URIs concerts occurring on Mondays

```
1 HTTP/1.1 200 OK
2 Date: Tue, 23 Aug 2011 15:43:28 GMT
3 Server: GlassFish Server Open Source Edition 3.1
4 Content-Type: application/rdf+xml
5 Vary: Accept-Encoding, User-Agent
```

And result is the description of the asked venue as RDF/XML (datatypes have been stripped for readability purpose):

Listing 6.6: Example of RDF/XML output

```
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
     xmlns:gr="http://purl.org/goodrelations/v1#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns="http://oxford-live.co.uk/ontology#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:wgs84="http://www.w3.org/2003/01/geo/wgs84_pos#">
  <EatPlace rdf:about="http://ol.filliau.com/resources/venue/vOYRY">
    <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#Venue"/>
    <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#French"/>
    <hasEatPlaceType rdf:resource="http://oxford-live.co.uk/ontology#Bar"/>
    <hasEatPlaceType rdf:resource="http://oxford-live.co.uk/ontology#Cafe"/>
    <hasEatPlaceType rdf:resource="http://oxford-live.co.uk/ontology#Deli"/>
    <hasEatPlaceType rdf:resource="http://oxford-live.co.uk/ontology#TakeAway"/>
    <hasIdentifier>vOYRY</hasIdentifier>
    <rdfs:label>Patisserie Valerie</rdfs:label>
    <wifiAccessAvailable>true</wifiAccessAvailable>
    <vcard:ADR rdf:parseType="Resource">
```

```
      <vcard:Street>90 High Street</vcard:Street>
      <vcard:Locality>Oxford</vcard:Locality>
      <vcard:Pcode>OX1 4BJ</vcard:Pcode>
    </vcard:ADR>
    <wgs84:long>-1.2531699</j.1:long>
    <wgs84:lat>51.7525561</j.1:lat>
    <owl:sameAs rdf:resource="http://linkedgeodata.org/triplify/node522986710"/>
    <owl:sameAs rdf:resource="http://www.dailyinfo.co.uk/reviews/venue/806/Patisserie_Valerie"/>
    <vcard:TEL rdf:parseType="Resource">
      <rdf:value>+44-1865-725415    </rdf:value>
    </vcard:TEL>
  </EatPlace>
</rdf:RDF>
```

7. Agent general design

7.1. Agent principle and responsibilities

Agents are independent applications that connect to the server with information retrieved from a data source. The agent has generally been made to analyse and retrieve information from one data source.

The server will never query the agent, but the agent is free to call the web service whenever it has information. It is entirely free to choose its method of retrieving information (it could also be – instead of retrieving data from the source – the source that would push information to the agent). This is a way to decentralize logic proper to a data source.

Some agents may only retrieve venues or both venues and events. Agents working with events must know the URI of the event before doing any operation (they can still search for a venue by its name and geo coordinates).

7.2. General architecture of an agent

As agents will use the RESTful web services of the server (described in section 6.4.2), a Java library has been realised that encapsulates all HTTP connections made to the server.

Agents are free to use or not this library, but it considerably reduces time of development as developers don't need to think about network communication (it acts as a black box, developers only use standard methods in their program). Written in Java, it is *de facto* ready to use for every language using the Java platform (Java, Scala, Groovy…).

Agents can also embed ontologies thanks to the Java library described in section 5.3.

An agent generally does three main operations in the following order:

1. retrieve and parse data from a data source
2. transform and sanitize data from a given data sources to a format suitable (RDF) to the server (this process is called *ontology alignment* as information will be expressed to fit the ontology defined in the application and described in chapter 5)
3. conversation with the server to create or update data

Conversation with the server is generally done in the order shown in figure 7.1. It shows the principle of informing the server of an *event*, but only the part about venue (which is mandatory as an event must include the URI of its associated venue) is detailed in the figure. It is the same steps for the creation of an event.

Agent first asks the server if a venue is known by its URI. If not, it asks the server if the venue is known by its name and its geo position. If it is known by name and address, the existing venue is updated to include the URI of the distant system as representing the same venue than in our system. The "worst" case is to create a new venue.

Once we have the URI of the venue, the process is the same for the event (first search by its URI and then by its name and date / time).

7.3. Agents built

Following research on suitable data sources done in section 2.2, some agents have been built.

7.3.1. DailyInfo

This agent is in charge of analysing the website DailyInfo [*DailyInfo*]. There is no API available, HTML has to be parsed. Jsoup [*JSoup website*], an HTML parser written in Java has been used. Both venues and events are contained – and interesting – in this website, two distincts parsers have been written for those two different types of pages.

HTML code being quite "irrespectful" of every convention or specification exisiting,

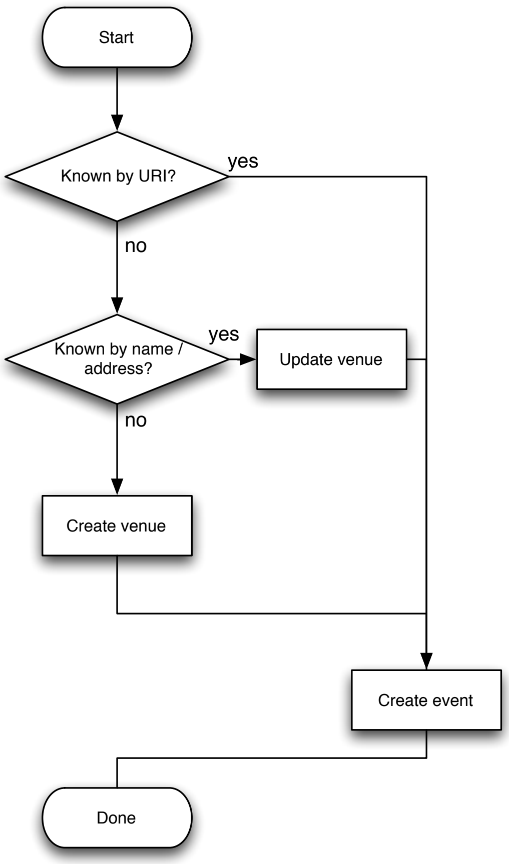


Figure 7.1.: Example of an agent trying to add a venue. Each diamond represents a query made to the server, each square an information pushed to the server.

parsing is quite complicated (information is not correctly structured). In addition to that, a lot of information seems to be contributed as free text. It is problematic for some tasks such as finding the time of an event. Indication of time is not always at the same place in the description of the event and format is not always the same. The Java library JChronic [*JChronic website*], specialized in "guessing" date and time in a string, has been used in an attempt to mitigate this issue.

To try to better understand the meaning of content in a page, one of the strategy used to determine the type of the venue (e.g. bar, pub...) was to fill some search parameters with values known to be correct and see what was retrieved (*deep web indexing* approach). This has been used to define types of venues and date of events.

7.3.2. Eventful

Eventful [*Eventful*] provides an API and a client library in Java that allowed the building of an agent only manipulating Java objects. Process has been quite straightforward being "just" an ontology alignment between the representation of Eventful and ours.

7.3.3. LinkedGeoData

LinkedGeoData [*LinkedGeoData*] dataset is provided as an RDF/NT file. The first step has been to mount the RDF file as a triple store to be able to query it efficiently. As this file contains basically every point contributed in OpenStreetMap everywhere in the world, it is quite heavy (more than 10 Go) and it has been necessary to filter this dataset and create a new RDF file containing only resources in Oxford. Listing 7.1 shows the SPARQL query that found all resources of type "amenity" contained in a box whose latitude and longitude corresponds to the minimum and maximum latitude and longitude of Oxford.

Listing 7.1: Query to filter LinkedGeoData dataset

```
PREFIX geo:<http://www.w3.org/2003/01/geo/wgs84_pos#>
SELECT ?o {
 ?o geo:lat ?lat .
 ?o geo:long ?lon .
 ?o <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://
     linkedgeodata.org/ontology/Amenity> .
 FILTER (?lat > 51.7028) .
 FILTER (?lat < 51.8165) .
 FILTER (?lon < -1.1797) .
 FILTER (?lon > -1.3386)}
```

This new file has then been processed in Java to create resources that match our ontology.

7.3.4. Oxford Talks

Oxford Talks [*University of Oxford - Oxford talks website*] publishes events of University of Oxford as vCal. An initial parsing of the HTML (with JSoup parser [*JSoup website*]) gives links to individual events that are then analysed with the help of the library iCal4j [*iCal4j website*].

The development of this agent has been discontinued. It appears that a lot of events where not *public* events (members of University of Oxford only) and that structure of postal addresses of events were difficult to analyse. Address field is only constituted of a string containing very (too much) precise information such as the room of the event which was leading to a creation of a lot of venues[1].

[1]Which is not incorrect depending of the notion of "venue", but it was not constant in the present case, as names were varying from an event to another.

8. Building an Augmented Reality application on top of the server

As a practical example of the scalability of the system, it has been decided to construct a basic augmented reality application to extend the platform to mobile devices. The aim of this application is to display results of the search of venues and events near the location of the user, on its smartphone.

8.1. What is "Augmented Reality"?

"Augmented reality" is the process of displaying information in real-time on top of a live view of the world. *Sensors* and *actuators* are used to add a new "layer" to the reality.

There is a wide range of devices, from smartphones (smartphone's camera) to specialized glasses or helmets and applications (helping navigation, displaying virtual objects on top of *real* objects to "improve" them. . .) available to do "augmented reality".

8.2. Layar – an "Augmented-reality browser"

Layar [*Layar website*] is a concrete, well-known application of augmented reality available for mobile devices (Android, iPhone. . .). Layar application provides an environment to display "points of interest" with filters based on criteria defined by the user.

The user has first to select a *layer* provided by a third-party (most of the time companies or brands) to display information based on its current location (this requires a compass and a GPS for best accuracy). A lot of layers are available to display information such as "cultural places around me", "hospitals and doctors near me". . .

Developers can easily build layers to display information by interfacing their system with the Layar platform.

8.3. Building an extension to plug into the Layar system

Regarding the localised nature of the work done with the prototype, it seemed interesting to interface the core server (described in section 6) and the Layar platform in order to be able to display venues and events near the user.

8.3.1. Interfacing the platform with Layar

The first thing to do is to create a developer account on Layar website [*Layar website*]. It is then possible to create a new layer and configure it. The main requirement is to define the URL of a RESTful resource on a server that Layar server will call (as described in figure 8.1). All requests done by a smartphone go through the Layar server, there is no direct link between the smartphone and the server managing the layer. It allows Layar to cache data.

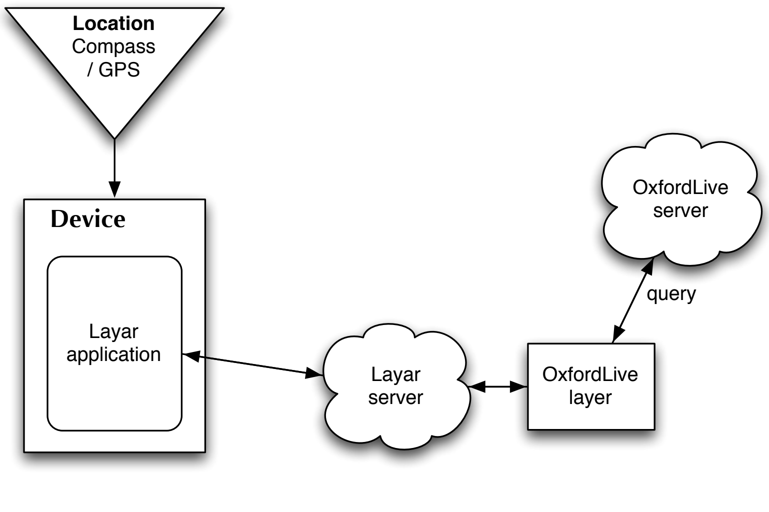


Figure 8.1.: Overview of the Layar architecture, “OxfordLive” represents here the server that will host the application.

The developer has to configure its service to match the specifications of the request and response that have to be made to the Layar server. The HTTP request realised by Layar includes parameters such as:

- latitude, longitude, radius, and accuracy of the user's location
- id, language and country of the user
- filters of the query (option "RADIOLIST", *v* indicates that it will filter on venues)

The following example shows a query realised by Layar server to the application server.

```
http://ol.filliau.com/resources/GetPOIs?lang=EN&countryCode=GB
&lon=-1.245165275&userId=dfe[...]c13&RADIOLIST=v&action=refresh
&version=6.0&radius=254&lat=51.750741975&layerName=oxfordlive&accuracy=75
```

Implemented RESTful web service is similar to other resources that have been described in section 6.4.2. Representation of data is done in JSON and respects a defined data model, both being conditions of the Layar service. Figure 8.2 describes the entities mandatory in the response of the server. Main entity is called *poi* (Point Of Interest) and describes – in our case – venues or events that will be displayed.

The initial implementation and working version of this service (creation of a new RESTful web service and Java objects to represent Layar response format) took around five hours. It shows the ability of the platform to add new services in a very short time.

8.3.2. Results

When lauching Layar on a smartphone and after selecting the developed layer, the user can now select "venues", "venues known to be open now" or "events occuring now" and filter per range around the current position of the user, as suggested in figure E.1.

After validating the query, the user will be able to browse results as "points of interest" displayed on a map (figure E.2b) or as a new *layer* to their camera (figure E.2a).

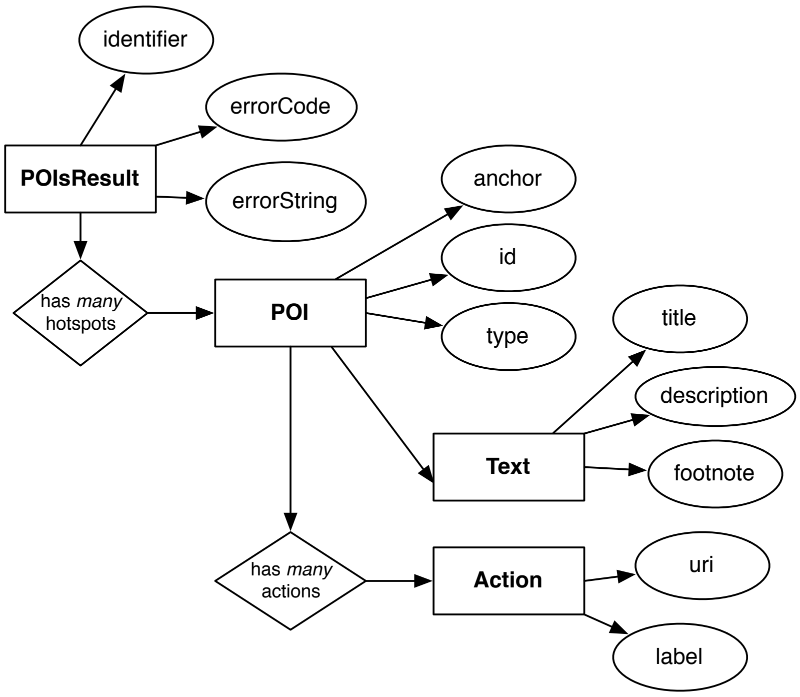


Figure 8.2.: Entity relationship diagram of the JSON representation of “points of interest”.

9. Evaluation

9.1. Evaluating the “wisdom of the crowds” dimension

This dimension will be evaluated in two different ways: active (testers filling a questionnaire) and passive (observation of the behaviour of testers).

9.1.1. Users test and questionnaire

This experiment is about getting users to test the prototype and then fill a questionnaire. They will first receive a short introduction to the system and its objectives and a link to try the prototype online. An online questionnaire (using Google Docs Forms feature) will propose users to evaluate both the idea and its implementation (the prototype).

The questionnaire will cover the following themes (full questions are available as appendix A):

- evaluation of the idea (objectives and motivation of the system)
- evaluation of the prototype and its features
- open questions to understand what they see on the prototype (based on what they are comparing the prototype to and what features they would like to see in addition to what has been implemented)

Considering the limited resources to publish the prototype online and the potential “illegal” data scraping, the testers will be limited to some known people familiar with Oxford.

9.1.2. Analysis of the use of the website

The behaviour of users will be analysed thanks to application logs. The following properties will be available for analysis:

- page views
- user's entered data in forms
- additions to the datastore
- errors

This should allow us to understand what parts of the application are the most used (and of interest for users). Evaluation of contribution will be done by watching additions to the datastore.

9.2. Evaluating the automatic gathering of data

Evaluating the work of agents feeding the server will be done by establishing metrics based on properties of agents:

- *time* to create an agent (if a lot of time is spent on programming an agent, it means that the source is difficult to analyse)
- *reliability* of the agent (number of errors in crawling, maybe due to changes in the source or incorrect information because of conflicts between sources)
- *usefulness* of the agent: number of new information retrieved by this agent (the more there is new information, the best it is, but just the fact that an information in one source corresponds to the same resource in another source is already useful to build a "comprehensive" view).

10. Results and Achievements

10.1. Results

10.1.1. Crowdsourcing evaluation

Results of questionnaire

Questionnaire has been run between August 12th and August 16th, 2011. It has received 15 answers. 12 out of 15 answers are students and are between 21 and 35 years old. 3 out of 15 answers are consultants, between 47 and 55.

Interest in the concept 87% of polled answer that they have difficulties to find local information on the internet, and 93% answer that they do see an interest in having a tool focused on their city. 80% of users see an added value in the prototype and all polled users say to be ready to use such a tool if it was available.

Evaluation of the prototype 73% of testers found the user interface intuitive and 27% found it confusing. *Searching* features are interesting for 67% of users and *contributing* features for 73%.

Tools to find local information Most cited sources to find local information are Google (and Google Maps, for search of venues) and DailyInfo.co.uk, mainly a directory of events and venues in Oxford.

Adding concepts to the tool Testers report that they would like to broaden types of venues (sport centres, cinema, accommodations) and add more details on ontology to find more fine-grained events (e.g. jazz concerts, not only concerts) and new types of events such as sporting events. Most cited new properties that they would like to see added are the availability of a vegetarian menu, availability of an outside garden[1], serving alcohol, ranking prices.

Testers also propose more "real-time" features to include current availability (if a venue is busy or if there is enough space for seating, tickets available for an event) and a "social layer" with *connection* to (existing) social network to see people recommendations or most frequented venues.

Adding features to the tool New features proposed by testers include a search by current position, notifications of new results of a query "pushed" to the user (email or RSS feed are suggested), the ability to refine a query after having initial results. Testers also propose to rate venues (without any precision on which type of rating).

Some users also noted their confusion because they had no results for a given query.

Results of analysis of users' behaviour

The analysis of access log shows which web pages have been the more viewed. It appears that the section "Search for eat and drink places" has been the more visited with 98 accesses, followed by the section "Search for events" (38 accesses) and finally "Search for venues", 26 visits. Pages to have details about individual venues and events have been viewed respectively 43 and 4 times. Forms to add new information (i.e. contribute opening hours, add an event, add a recurring event) have been *viewed*[2] respectively 7, 3 and 2 times.

No particular error has been detected during the length of the experimentation.

Following comments of testers, it appears that they have been confused by the field "Where?". Some people entered postcodes, streets and "well-known places" (e.g. Sheldonian theatre). Some comments suggest that – as the textbox was empty by default – they didn't dare to write everything they wanted to write, and suggest as a new feature to be able to search per venue or area, street number.

Basically, all possibilities wanted as "additional features" were already working. Also, some users made some bad spelling (such as writing 0X4, with a zero instead of

[1]Ontology is ready for this property but there is a lack of information available on that subject.
[2]Viewing the form does not mean creating new information.

a “O”) which gave them no results.

Other comments show that the default search filtering on venues opened at a given time (set by default at “now”) has been misunderstood. Indeed, a lot of users were running queries that should have gave them some results…if every venue had its opening hours filled (very few venues have them filled, as covered in section 10.1.2). A label was indicating that search will be done on venues *known* to be open, but it seems that it has been ignored.

Results of users’ contributions in the prototype

It appears that only three opening hours for a venue have been contributed. No event nor recurring event has been contributed.

It is interesting to notice that contributed opening hours are incorrect because represented as a human would write it, with a venue closing at 3 o’clock in the morning[3]. Although form should have been validated, it shows the difference of approach between a human user (able to understand a closing at “3 o’clock in the morning”) and rigorous knowledge expected in a knowledge-base.

10.1.2. Results of automated gathering of information

Agents have been running in the following order:

1. LinkedGeoData (LGD): quality of data and especially geo points appears to be excellent so it has been decided to use if as a reference, a basis for venues.
2. OpeningTimes
3. Eventful
4. DailyInfo

Following table summarizes how much venues and events have been created and updated. An approximative time taken by the agent to complete its work is also displayed. “Hours of work” represent an amount of time taken to realise the agent (excluding time spent to learn how to use development tools).

[3]Contributed hours are the following: “07:30:00 03:00:00 Friday”, with a time format of 24 hours. In reality, system was expecting a closing at midnight and a new opening on a Saturday until 03:00.

Data source	*Created venues*	*Updated venues*	*Created events*	*Updated events*	*Time running*	*Hours of work*
LGD	544	0	n/a	n/a	4 min.	5 h.
Op.Times	22	8	n/a	n/a	3 min.	3 h.
Eventful	37	15	52	0	8 min.	3 h.
DailyInfo	339	210	250	0	60 min.	8 h.

No particular *error* has been detected during agents work. A lot of imprecise data seems to be present. Unfortunately it has not been possible to quantify this because of the mashing up of data sources – finding which data are incorrect is a manual work that need a good knowledge of Oxford. It is also not possible to compare our "result" with reference data because it is an aggregation of data. Globally, main problems are:

- duplicate venues because of:
  - imprecise address, if the result of the geocoding is too far compared to the first venue, then a second venue is created
  - imprecise name, if the name is not well defined it will lead to the creation of a new venue (e.g. if a venue A exists with name "Santorini Greek Restaurant" in a datasource B, and the same venue A is identified as "Santorini" in a datasource C, it may[4] create a second venue)
- missing venues because of a not unique URI: it has been assumed that every link (URL) would represent uniquely one venue, but it is sometimes not the case. For instance, *Mortons* has many venues (franchises) in Oxford, but all venues have their URI defined as `http://www.mortonsatwork.co.uk`, leading to an inability to have all venues displayed correctly. This case is not common but it shows the importance of well-defined URIs to represent "things".
- venue inside a venue also causes problem. The case of *Costa Coffee* inside *Waterstones* – with the same geo point, and part of the same name led to the creation of only one venue.

Regarding time spent to develop each agent, there is no *a posteriori* regret because process has finally been quite straightforward and agent where time has been most spent (DailyInfo) has so much information that it was still worthwhile. Quality of agents

[4]It depends on the distance between the strings, calculated by full-text search engine Lucene.

could be improved (particularly DailyInfo) with different approaches to understand natural languages.

10.1.3. Summary of achievements

A running prototype has been successfully implemented with four running agents. Test with real users has run smoothly and useful results came out of the evaluation.

10.2. Critical assessment

10.2.1. Critical review of work

Most annoying part of the work has been the impossibility to try some technologies for agents (such as GRDDL, all described in section 2.1) because of a lack of suitable data sources (datasources were mostly HTML or APIs, so the work of agents was not only a *transformation* but also a cleaning and sanitize operation).

10.2.2. Review of prototype

Server design

Performance Globally, server application appears to be slow. Caching of data could be a solution but it also seems that the merging of data between the triple store and Lucene indexes takes a lot of time, as noted as a potential risk in section 2.4 and in [Delbru (2010)].

Alternative design of input *A posteriori*, design of input data into the server may not be efficient as it supposes a lot of back and forth between agents and server. Also, the process being synchronous it supposes to wait the response of the server to continue its process. Alternative design could be an agent being able to send all data that it contains to the server ("*here is everything I know*"), server doing much of the process. This is less respectful of a RESTful architecture, and agent role is reduced but it may be more efficient. It could also be interesting to have asynchronous processing using messaging (accepting content but not creating it immediately) but it means that there is no URI created and that content is not available directly to be reused (e.g. creation of a venue to create an event).

Lack of features Some features such as the versioning of data (as a semantic wiki) have not been implemented due to a lack of time and design (although the most easy way to implement it would be to use the RDF changeset vocabulary).

Security No security features have been implemented (users management in the web application, agents management...). In particular, agents should be able to register themselves in the system in order to be authorised to push data into our system. It doesn't appear safe not to have any authentication mechanism as it could encourage spammers or malicious users.

Data integrity Process of adding (and updating) data should perform much more data integrity check in order to check if there is no duplicated data. If an user, for example, try to insert an event, this component should check if an event with a similar date and venue already exists in our data store. In some case, it may be necessary to make some tradeoffs if data are imprecise.

Web application

Ergonomic Evaluation shows that user interface doesn't appear very intuitive to users. Auto-completion of fields (as suggested in background review) may help the user to express its query (e.g. when adding an event, if the user try to find an associated venue, a "smart" auto-complete service should propose him the venue).

Mobile Regarding the localised nature of data, it would have been interesting to add mobile views to the web application in order to be able to use it with smartphones.

11. Conclusion

11.1. Lessons learnt

11.1.1. Value of data

During this dissertation, the high value of data has been demonstrated. Studying terms of use of various providers and running into a rebuff to get authorisation for academic research purpose, it shows how sensitive are people regarding their data silo.

On the long view, what is the benefit of holding data without giving the ability to exhibit them (*open data*)? Exposing data does not necessarily mean lose ownership but could lead to a greater audience viewing them.

11.1.2. Semantic Web architecture vs. Object Oriented

Building Semantic Web applications is a different approach of object-oriented programming. Even if the temptation to apply Object-Oriented programming pattern is there, developer must be careful as the expressivity of RDF is much higher than a model expressed with objects. Applying object-oriented patterns may reduce the usefulness of using Semantic web technologies.

New frameworks such as Apache Clerezza [*Apache Clerezza website*] aim to help developers to "think semantic" and would be interesting to look at in the future.

11.1.3. Building ontologies

Although there is not one good way of representing knowledge (i.e. building an ontology), it must be carefully planned as it is then difficult to migrate data. Ontology versioning is a subject that would need more investigation. It is also necessary to make tradeoffs re-using existing ontologies because it can lead to an overload of possibilities to describe information.

Building one ontology to describe the domain and do *ontology alignment* in agents appears the most convenient (and maybe neat too), but is it the better approach to value data? Having one ontology per agent (per data source) may be more respectful of data (and would allow to appreciate much more the richness of them), although it would lower the extensibility of the system as links between ontologies must be done manually.

11.1.4. Interest in mashing-up local data

Compared to tools focusing on a specific subject (e.g. conferences everywhere in the world), having a tool focused on a local area, on various subjects, appears to be a complementary approach adding value for local users.

Some similar and unrelated projects are emerging such as the proposition in [*Oxford Fairtrade Hack-day*] of a tool to map local fairtrade activity in Oxford.

11.2. Suggestions for further work

User interfaces

It seems challenging to be able to build "simple" interfaces that make the most of semantic web technologies (the aim being that underlying technologies are not visible to the end-user). Exploring content using faceted search on top of a triple store, or tools to help a user refine his results (or its query) would be interesting subjects to study.

Rating

Rating the quality of entered data (and by extension of users contributing data) would allow to establish levels of confidence in users.

Also, offering the ability to users to rank venues and events with various criteria would help users to find which venues are the most "interesting" based on those criteria.

Social network

As proposed by testers of the prototype, the integration with social networks would allow users to see recommendations of their friends and be able to do “check-ins” in venues (or events). This could also allow us to be able to “track them down” to discover patterns of users moving in the city and understand what is popular.

Recommendation

Based on the two precedent propositions, how could it be possible to build a recommendation engine based on the profile of the user and rated information?

Engaging users

Engaging users to contribute to the platform also seems to be a challenge. Approaches using games (winning points for doing actions for instance), are becoming more and more present nowadays, it would be interesting to study how it could be used to favour users to contribute.

Connection with the “real”-world

Due to the localised nature of the work, how would it be possible to connect the “virtual” world to the “real” world? RFiD (Near Field Communication), QRCode would be interesting technologies to investigate in order to make links between a web-based system and real objects.

Extending the data model

It would also be interesting to extend the data model with much more precisions about existing types and add new types.

How could it be possible to *map communities* of a given area? People are indeed members of many social communities, and finding / discovering those communities – that are often niches – may interest some people (especially newcomers, for instance there is a lot of technology communities in Oxford that it is hard to find at first sight).

How could it be possible to map the availability of products (goods) in the city? Ontology GoodRelations [*GoodRelations homepage*] has some features to cover this and "ProductDB" [*ProductDB website*] aims to interconnect information about products (using linked data).

People could also be able to contribute to the schema and add properties (such as a "fairtrade property" as suggested in [*Oxford Fairtrade Hack-day*]).

It would also be interesting to represent more complex recurrences and be able to annotate the validity in time of an information (e.g. opening hours valid only for 2011).

Finally, considering this prototype, city is a unit, but it is not a limit. We could imagine to build a grid of instances and be able to query multiple instances at the same time, giving the ability to find information on more than one local unit (more than one city).

A. Questionnaire

Complete questions of the questionnaire.

You

What is your job?

What is your industry?

What is your age?

Your interest in the concept

Do you usually find local information easily (on the internet)?

Yes - very easily / Yes - but scattered into many websites / No - it is difficult / No - I often cannot find information

Do you find an interest of having a tool focused on an area (your city)?

Yes - strong interest / Yes - little interest / No - not at all

Do you see the added value compared to existing tools?

Yes / No

Would you use a tool like this prototype if it was available?

Yes - on a regular (frequent) basis/ Yes - infrequently / No

Evaluation of the prototype

Do you find the user interface intuitive?

Yes - very intuitive / Yes - intuitive / No - confusing / No - not at all

Do you find SEARCH features of the prototype interesting?

Yes / No

Do you find CONTRIBUTE features of the prototype interesting?

Yes / No

Open questions

What tools do you use to find local information?

What other "concepts" (like events and venues) or "properties" (like wifi available, disability access...) would you like to see in this kind of tool?

What additional features would you like to see in such a tool?

Additional comments

B. Ontology

Listing B.1: Ontology

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://oxford-live.co.uk/ontology#"
     xml:base="http://oxford-live.co.uk/ontology"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about="http://oxford-live.co.uk/ontology#"/>

    <!--
    // Datatypes
     -->

    <rdfs:Datatype rdf:about="&xsd;time"/>

    <!--
    // Object Properties
     -->

    <owl:ObjectProperty rdf:about="http://oxford-live.co.uk/ontology#
        hasEatPlaceType">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
        <rdfs:range rdf:resource="http://oxford-live.co.uk/ontology#EatPlaceType"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://oxford-live.co.uk/ontology#hasVenue">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#Event"/>
        <rdfs:range rdf:resource="http://oxford-live.co.uk/ontology#Venue"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="http://oxford-live.co.uk/ontology#occursOn">
        <rdfs:comment>Day of a recurring event.</rdfs:comment>
        <rdfs:range rdf:resource="http://oxford-live.co.uk/ontology#Day"/>
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#Event"/>
    </owl:ObjectProperty>

    <!--
    // Data properties
     -->

    <owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#beginAt">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#Event"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#endAt">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#Event"/>
        <rdfs:range rdf:resource="&xsd;dateTime"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#fireAvailable">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#Venue"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    gardenAvailable">
    <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
        Venue"/>
    <rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    hasGlutenFreeAvailable"/>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    hasHandicapAccess"/>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    hasIdentifier">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:label>Unique identifier of the resource</rdfs:label>
    <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
        Event"/>
    <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
        Venue"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    hasPrice">
    <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
        Event"/>
    <rdfs:range rdf:resource="&xsd;double"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    hasUsualHour">
    <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
        Event"/>
    <rdfs:range rdf:resource="&xsd;time"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
    name">
```

```
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
            Venue"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
        title">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
        url">
        <rdfs:range rdf:resource="&xsd;anyURI"/>
        <rdfs:domain rdf:resource="&owl;Thing"/>
    </owl:DatatypeProperty>

    <owl:DatatypeProperty rdf:about="http://oxford-live.co.uk/ontology#
        wifiAccessAvailable">
        <rdfs:domain rdf:resource="http://oxford-live.co.uk/ontology#
            Venue"/>
        <rdfs:range rdf:resource="&xsd;boolean"/>
    </owl:DatatypeProperty>

    <!--
    // Classes
     -->

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#American">
        <rdfs:label>American</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Art">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Asian">
        <rdfs:label>Asian</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Books">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#British">
        <rdfs:label>British</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            European"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Caribbean">
        <rdfs:label>Caribbean</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Chinese">
        <rdfs:label>Chinese</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Asian"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Cinema">
        <rdfs:label>Cinema</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            LeisurePlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Concert">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Conference">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Day"/>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#EatPlace">
        <rdfs:label>Everything</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Venue"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#EatPlaceType"
        />

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#European">
        <rdfs:label>European</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Event">
        <rdfs:label>Event</rdfs:label>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Film">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#French">
        <rdfs:label>French</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            European"/>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Mediterranean"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Greek">
        <rdfs:label>Greek</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            European"/>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Mediterranean"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Indian">
```

```
        <rdfs:label>Indian</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Asian"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Italian">
        <rdfs:label>Italian</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#European"/>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Mediterranean"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Jamaican">
        <rdfs:label>Jamaican</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Caribbean"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Japanese">
        <rdfs:label>Japanese</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Asian"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Kebab">
        <rdfs:label>Kebab</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Oriental"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Lebanese">
        <rdfs:label>Lebanese</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Mediterranean"/>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#Oriental"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#LeisurePlace">
        <rdfs:label>Leisure place</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Venue"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Mediterranean
        ">
        <rdfs:label>Mediterranean</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Mexican">
        <rdfs:label>Mexican</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            American"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#MiddleEast">
        <rdfs:label>Middle East</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Oriental"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Oriental">
        <rdfs:label>Oriental</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#
        PerformingArts">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Quiz">
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Event"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Russian">
        <rdfs:label>Russian</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlace"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Shop">
        <rdfs:label>Shop</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Venue"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Swiss">
        <rdfs:label>Swiss</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            European"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Thai">
        <rdfs:label>Thai</rdfs:label>
        <rdfs:subClassOf rdf:resource="http://oxford-live.co.uk/ontology#
            Asian"/>
    </owl:Class>

    <owl:Class rdf:about="http://oxford-live.co.uk/ontology#Venue">
        <rdfs:label>Venue</rdfs:label>
    </owl:Class>

    <!--
    // Individuals
     -->

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#Bar
        ">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Bar</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        Cafe">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Cafe</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        Deli">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Deli</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        Friday">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#Day"/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        HomeDelivery">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Home delivery</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        Monday">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#Day"/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#Pub
        ">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Pub</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        Restaurant">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Restaurant</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about="http://oxford-live.co.uk/ontology#
        SandwichBar">
        <rdf:type rdf:resource="http://oxford-live.co.uk/ontology#
            EatPlaceType"/>
        <rdfs:label>Sandwich bar</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#Saturday”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#Day”/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#Sunday”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#Day”/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#TakeAway”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#EatPlaceType”/>
        <rdfs:label>Take away</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#Thursday”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#Day”/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#Tuesday”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#Day”/>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#Van”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#EatPlaceType”/>
        <rdfs:label>Van</rdfs:label>
    </owl:NamedIndividual>

    <owl:NamedIndividual rdf:about=”http://oxford-live.co.uk/ontology#Wednesday”>
        <rdf:type rdf:resource=”http://oxford-live.co.uk/ontology#Day”/>
    </owl:NamedIndividual>
</rdf:RDF>
```

C. Visualisation of ontology classes

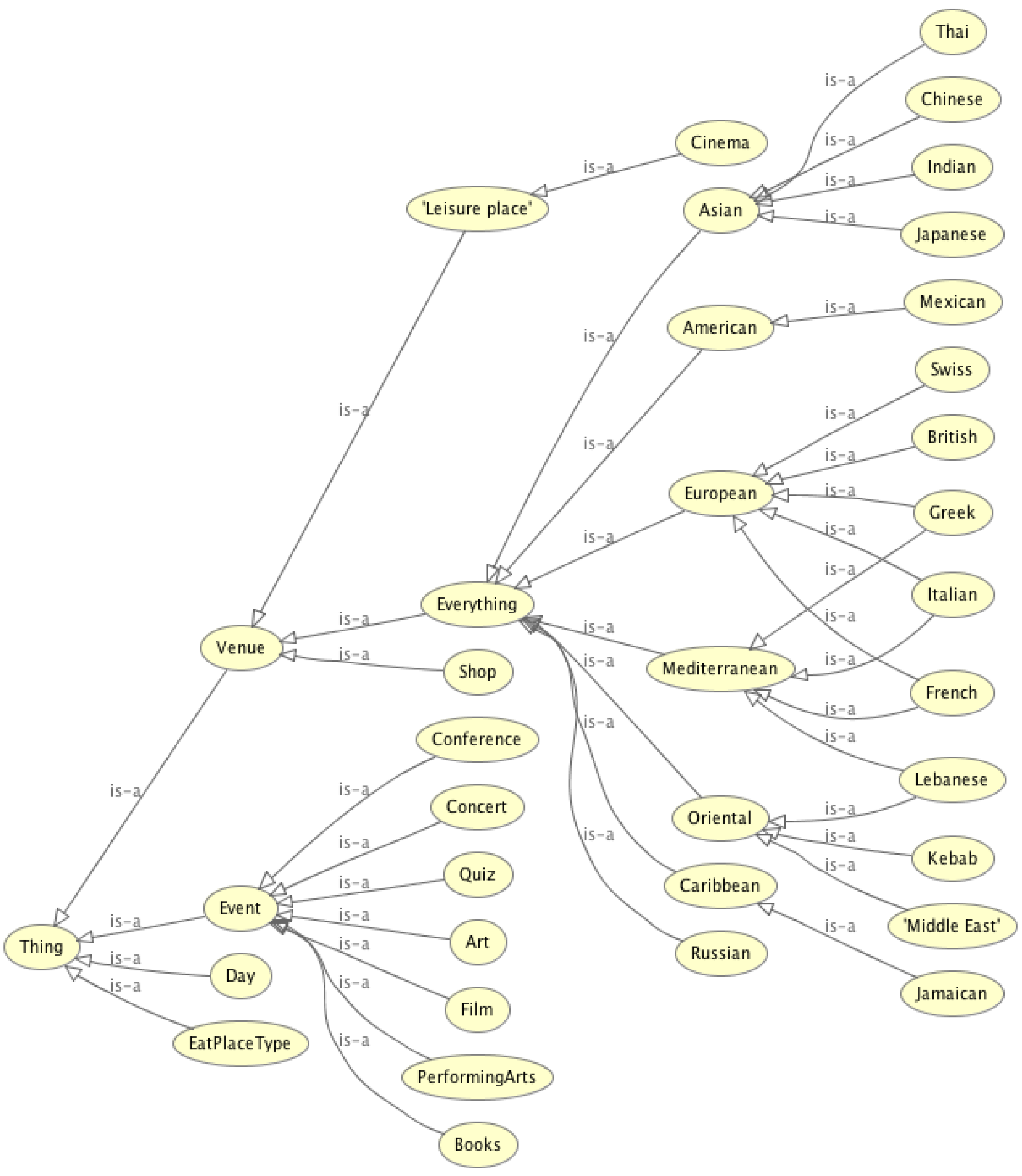
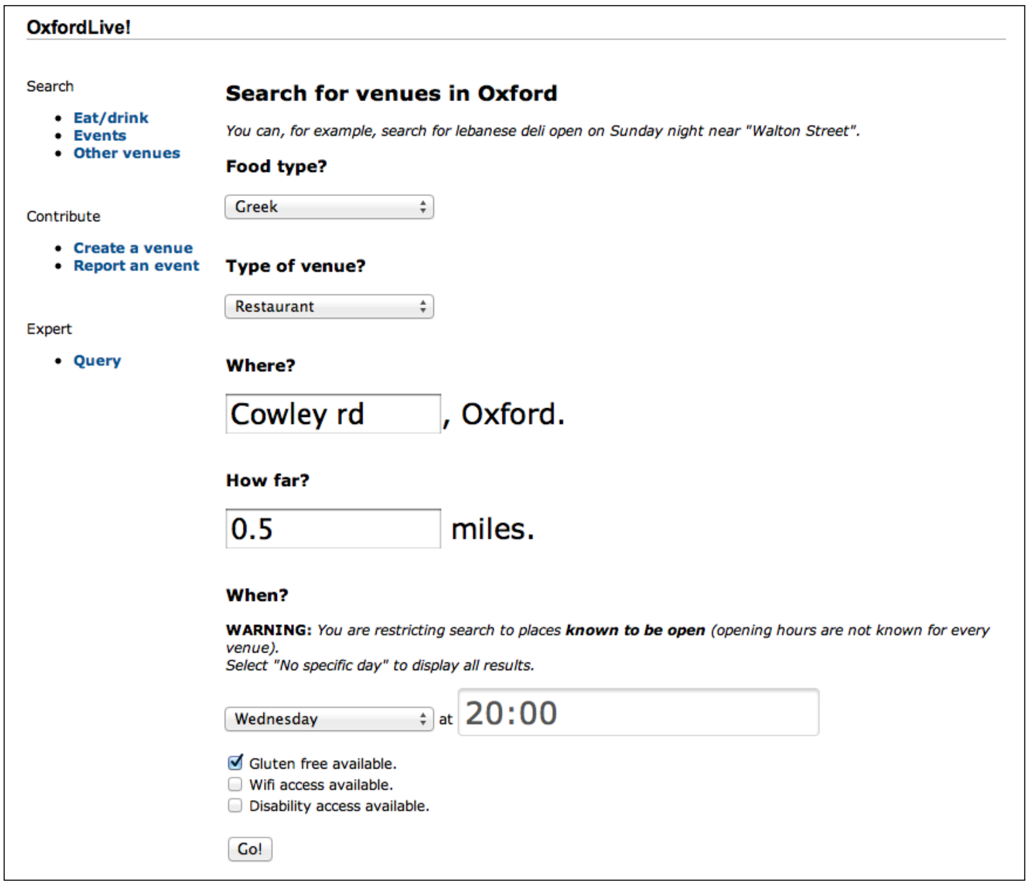


Figure C.1.: OWL classes

D. Web application screenshots



OxfordLive!

Search

- Eat/drink
- Events
- Other venues

Contribute

- Create a venue
- Report an event

Expert

- Query

Search for venues in Oxford

You can, for example, search for lebanese deli open on Sunday night near "Walton Street".

Food type?

Greek

Type of venue?

Restaurant

Where?

Cowley rd, Oxford.

How far?

0.5 miles.

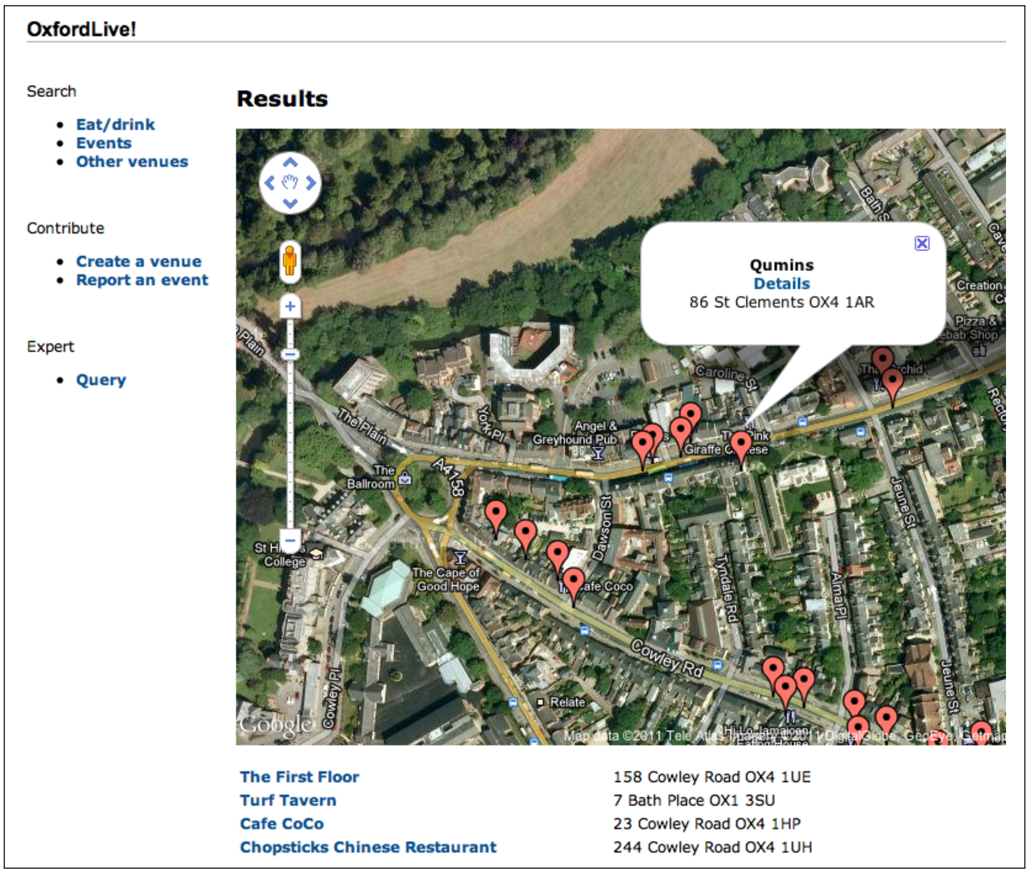
When?

WARNING: *You are restricting search to places* ***known to be open*** *(opening hours are not known for every venue).*
Select "No specific day" to display all results.

Wednesday at 20:00

- [✓] Gluten free available.
- [ ] Wifi access available.
- [ ] Disability access available.

Figure D.1.: Form to search for a place to eat


OxfordLive!

Search

- Eat/drink
- Events
- Other venues

Contribute

- Create a venue
- Report an event

Expert

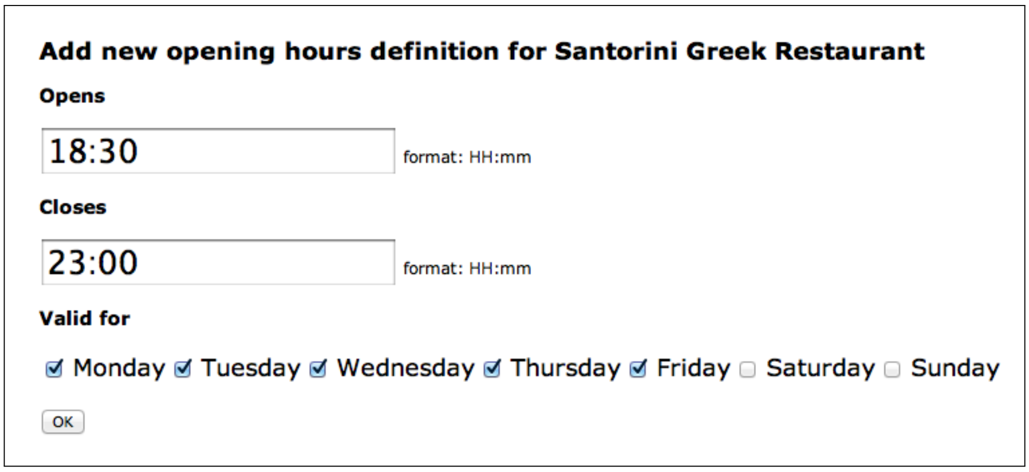
- Query

Results

Qumins
Details
86 St Clements OX4 1AR

The First Floor	158 Cowley Road OX4 1UE
Turf Tavern	7 Bath Place OX1 3SU
Cafe CoCo	23 Cowley Road OX4 1HP
Chopsticks Chinese Restaurant	244 Cowley Road OX4 1UH

Figure D.2.: Search results displayed as map and list


Add new opening hours definition for Santorini Greek Restaurant

Opens

18:30 format: HH:mm

Closes

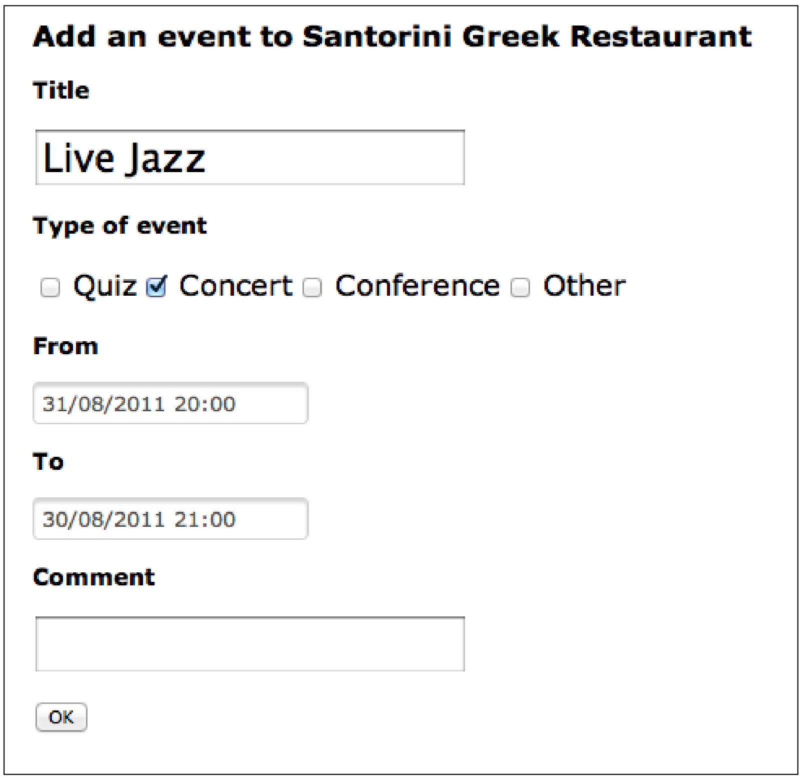
23:00 format: HH:mm

Valid for

[✓ Monday] [✓ Tuesday] [✓ Wednesday] [✓ Thursday] [✓ Friday] Saturday Sunday

OK

Figure D.3.: Adding opening hours to a venue


Add an event to Santorini Greek Restaurant

Title

Live Jazz

Type of event

☐ Quiz ☑ Concert ☐ Conference ☐ Other

From

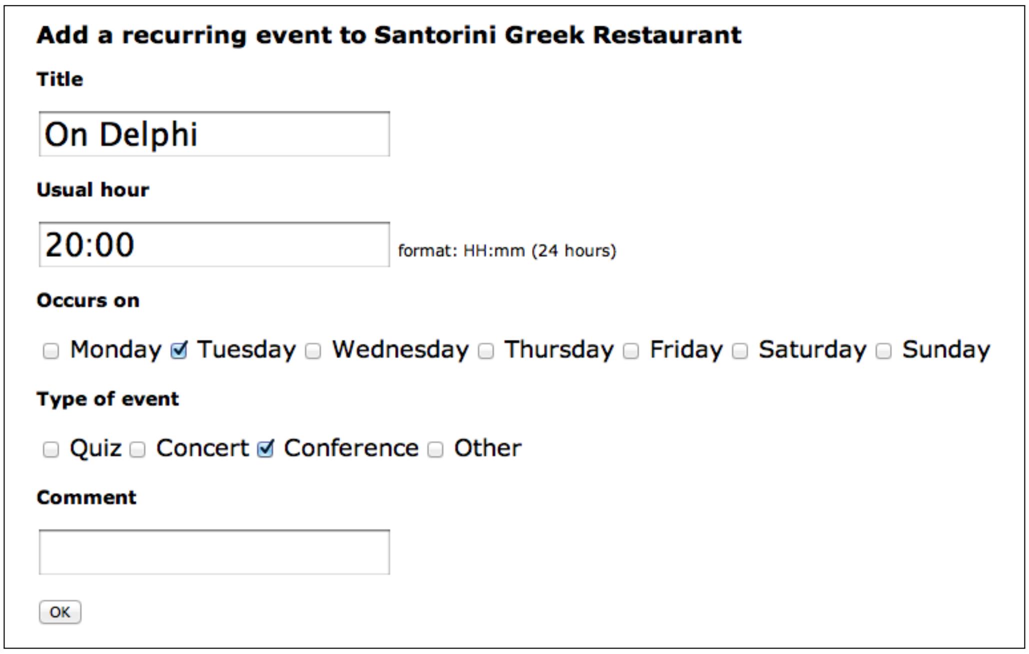
31/08/2011 20:00

To

30/08/2011 21:00

Comment

Figure D.4.: Adding a one-time event to a venue


Add a recurring event to Santorini Greek Restaurant

Title

On Delphi

Usual hour

20:00 format: HH:mm (24 hours)

Occurs on

☐ Monday ☑ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday ☐ Sunday

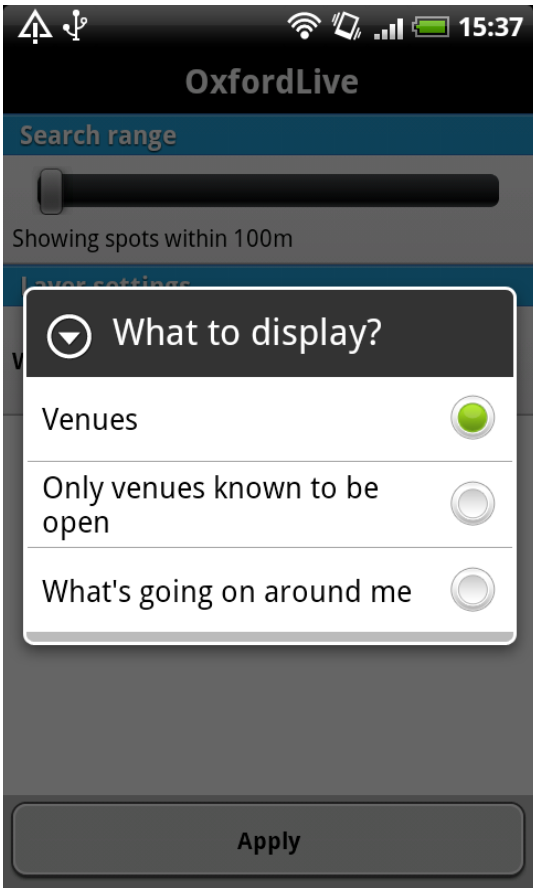
Type of event

☐ Quiz ☐ Concert ☑ Conference ☐ Other

Comment

Figure D.5.: Adding a recurring event to a venue

E. Layar screenshots


15:37

OxfordLive

Search range

Showing spots within 100m

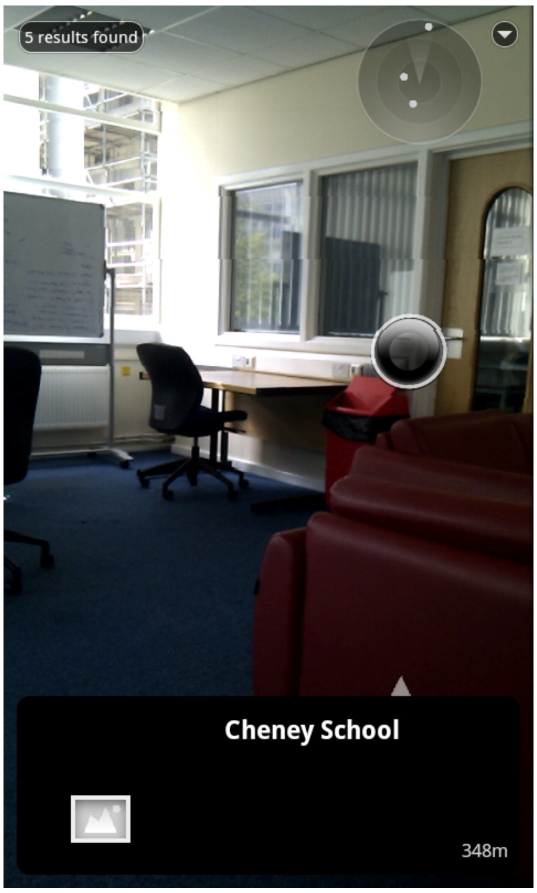
What to display?

Venues [✓]

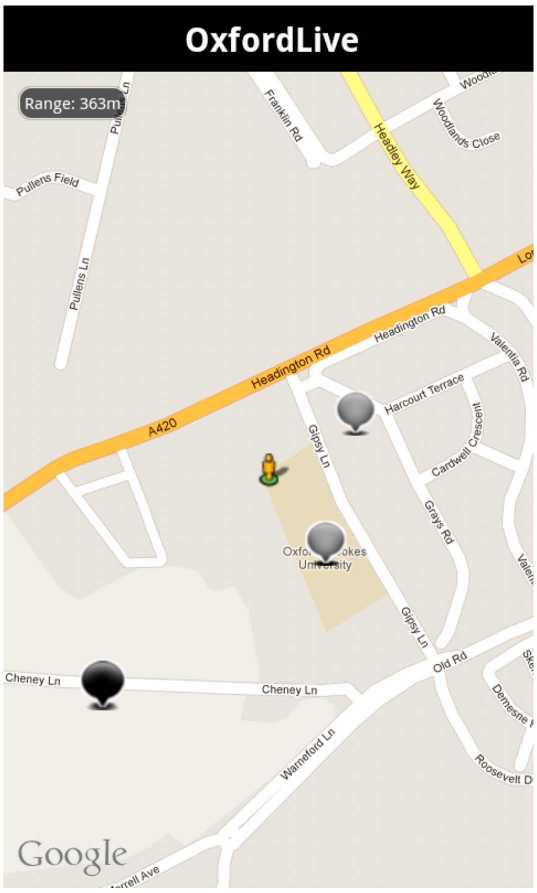
Only venues known to be open

What's going on around me

Figure E.1.: Layar Search view



(a) Camera view. Black bubble on top of the "live view" shows the direction of "Cheney school".

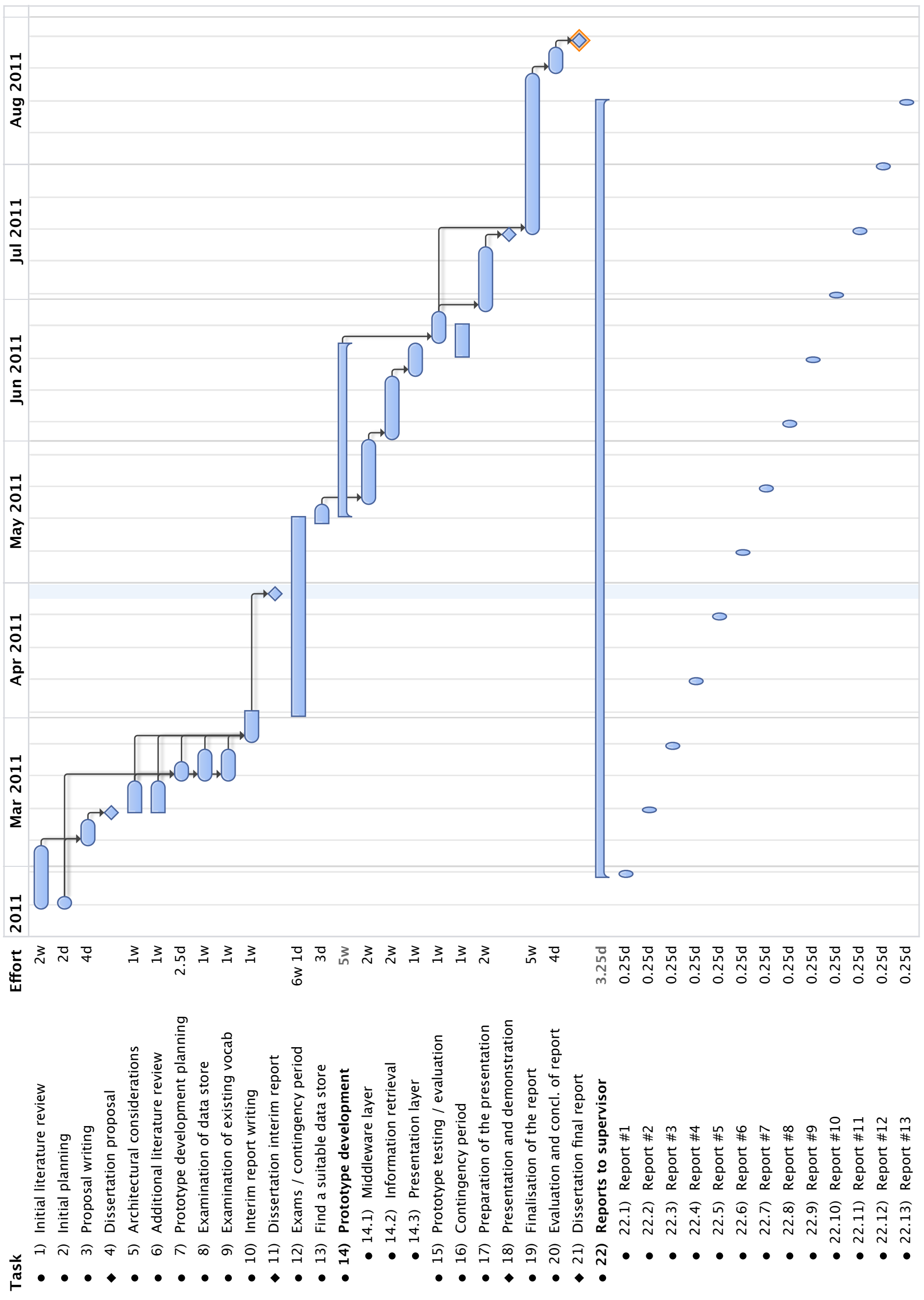


(b) Map view showing results near the Headington campus of Oxford Brookes University

Figure E.2.: Example of results

F. Project plan

Following specifications and design, the following plan has been realised. It covers the development of the prototype and the realisation of the final report.



Task	Effort
● 1) Initial literature review	2w
● 2) Initial planning	2d
● 3) Proposal writing	4d
◆ 4) Dissertation proposal	
● 5) Architectural considerations	1w
● 6) Additional literature review	1w
● 7) Prototype development planning	2.5d
● 8) Examination of data store	1w
● 9) Examination of existing vocab	1w
● 10) Interim report writing	1w
◆ 11) Dissertation interim report	
● 12) Exams / contingency period	6w 1d
● 13) Find a suitable data store	3d
● **14) Prototype development**	5w
● 14.1) Middleware layer	2w
● 14.2) Information retrieval	2w
● 14.3) Presentation layer	1w
● 15) Prototype testing / evaluation	1w
● 16) Contingency period	1w
● 17) Preparation of the presentation	2w
◆ 18) Presentation and demonstration	
● 19) Finalisation of the report	5w
● 20) Evaluation and concl. of report	4d
◆ 21) Dissertation final report	
● **22) Reports to supervisor**	3.25d
● 22.1) Report #1	0.25d
● 22.2) Report #2	0.25d
● 22.3) Report #3	0.25d
● 22.4) Report #4	0.25d
● 22.5) Report #5	0.25d
● 22.6) Report #6	0.25d
● 22.7) Report #7	0.25d
● 22.8) Report #8	0.25d
● 22.9) Report #9	0.25d
● 22.10) Report #10	0.25d
● 22.11) Report #11	0.25d
● 22.12) Report #12	0.25d
● 22.13) Report #13	0.25d

Figure F.1.: GANTT chart of the project

Glossary

API Application Programming Interface. 23, 58, 60, 73

HTML HyperText Markup Language. 52, 53, 58, 60, 73

HTTP (Hypertext Transfer Protocol), "a networking protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web." (Source: *Wikipedia on HTTP*). 38, 53, 54, 57, 65

JSON JavaScript Object Notation. 24, 54, 65, 66

Linked Data "describes a method of publishing structured data so that it can be interlinked and become more useful. It builds upon standard Web technologies such as HTTP and URIs, but rather than using them to serve web pages for human readers, it extends them to share information in a way that can be read automatically by computers. This enables data from different sources to be connected and queried" (Source: *Wikipedia on Linked Data*). 24, 33, 53, 78

Ontology "formally represents knowledge as a set of concepts within a domain, and the relationships between those concepts. It can be used to reason about the entities within that domain, and may be used to describe the domain." (Source: *Wikipedia on ontology*). 8, 10, 18, 26, 27, 32, 40, 41, 43, 44, 52, 70, 75, 76, 78, 103

OWL "Web Ontology Language is a family of knowledge representation languages for authoring ontologies." (Source: *Wikipedia on OWL*). 26, 28, 32, 33, 41, 42

RDF "similar to classic conceptual modeling approaches such as Entity-Relationship or Class diagrams, as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions. These expressions are known as triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one way to represent the notion ”The sky has the color blue” in RDF is as the triple: a subject denoting “the sky”, a predicate denoting “has the color”, and an object denoting “blue”. RDF is an abstract model with several serialization formats (i.e. file formats), and so the particular way in which a resource or triple is encoded varies from format to format” (Source: “Wikipedia on RDF”). 15–17, 28, 32–34, 42, 47, 53, 54, 58, 60, 74, 75, 104

RDFa “Resource Description Framework – in – attributes [. . .] adds a set of attribute-level extensions to XHTML for embedding rich metadata within Web documents” (Source: Wikipedia). 17

RDFS “RDF Schema is a set of classes with certain properties using [RDF], providing basic elements for the description of ontologies, otherwise called RDF vocabularies, intended to structure RDF resources.” (Source: *Wikipedia on RDFS*). 29, 32, 41, 42, 104

REST “Representational State Transfer is a style of software architecture for distributed hypermedia systems such as the World Wide Web.” (Source: *Wikipedia on REST*). 104

RESTful An application is RESTful when it satisfies REST conditions.. 23, 45, 51, 53, 57, 64, 65, 73

Semantic wiki Wiki with structured knowledge, may use technologies of the Semantic web.. 18, 74

SKOS (Simple Knowledge Organization System) is an ontology built with RDFS to represent taxonomies.. 19, 42

SPARQL Main query language of the Semantic Web. Equivalent of SQL for relational databases.. 16, 20, 24, 29, 30, 32, 45–49, 51, 53, 54, 60

SPARQL endpoint “enables users (human or other) to query a knowledge base via the SPARQL language. Results are typically returned in one or more machine-processable formats. Therefore, a SPARQL endpoint is mostly conceived as a machine-friendly interface towards a knowledge base.” (Source: *SPARQL End-*

point definition). 24, 53, 54

Triple Store Storage of RDF graph. 28, 32, 43, 46–48, 73, 76

Urban Computing “emerging field of study that focuses on the use of technology in public environments such as cities [...] It also studies the interaction between humans and such environments, which is becoming increasingly common as access to computing devices extends beyond home and office” (Source: *Wikipedia on Urban Computing*). 15

URI “Uniform Resource Identifier is a string of characters used to identify a name or a resource on the Internet.” (Source: *Wikipedia on URI*). 33, 36, 48–51, 55, 57, 58, 72, 74, 103, 105

URL “Uniform Resource Locator or Universal Resource Locator. It is a character string that specifies where a known resource is available on the Internet and the mechanism for retrieving it. A URL is technically a type of URI but in many technical documents and verbal discussions URL is often used as a synonym for URI” (Source: *Wikipedia on URL*). 54, 72, 105

References

Aart, Chris van, Bob Wielinga, and Willem van Hage (2010). "Mobile cultural heritage guide: location-aware semantic search". In: *Knowledge Engineering and Management by the Masses*. Ed. by Philipp Cimiano and H. Pinto. Springer Berlin / Heidelberg, pp. 257–271. DOI: 10.1007/978-3-642-16438-5_18. URL: http://www.springerlink.com/index/FW15R24305253H71.pdfhttp://dx.doi.org/10.1007/978-3-642-16438-5_18.

Ali, M.I. et al. (2011). "On Integrating Data Services Using Data Mashups". In: *infosys.tuwien.ac.at*, pp. 5–8. URL: http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:On+Integrating+Data+Services+Using+Data+Mashups\#0.

Apache Clerezza website. Accessed on 28/08/2011. URL: http://incubator.apache.org/clerezza/.

Auer, S., Jens Lehmann, and Sebastian Hellmann (2009). "LinkedGeoData: Adding a spatial dimension to the Web of Data". In: *The Semantic Web-ISWC 2009*, pp. 731–746. URL: http://www.springerlink.com/index/J63221026432X374.pdf.

Auer, Sören et al. (2009). "Triplify: light-weight linked data publication from relational databases". In: *Proceedings of the 18th international conference on World wide web.* ACM, pp. 621–630. URL: http://portal.acm.org/citation.cfm?id=1526793.

Babitski, Grigori et al. (2011). "SoKNOS Using Semantic Technologies in Disaster Management Software". In: *8th Extended Semantic Web Conference (ESWC2011).* URL: http://www.heikopaulheim.com/documents/eswc_2011.pdf.

Basic Geo Vocabulary specification. Accessed on 05/06/2011. URL: http://www.w3.org/2003/01/geo/.

Bizer, Chris and Andreas Schultz (2011). *Berlin SPARQL Benchmark.* Accessed on 29/06/2011. URL: http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/results/V6/index.html.

Bootstrapping the Semantic Web with GRDDL, Microformats, and RDFa. Accessed on 27/08/2011. URL: `http://www-sop.inria.fr/acacia/personnel/Fabien.Gandon/tmp/grddl/grddl-introduction-v3/`.

Buffa, M et al. (2008). "SweetWiki: A semantic wiki". In: *Web Semantics Science Services and Agents on the World Wide Web* 6.1, pp. 84–97. ISSN: 15708268. DOI: 10.1016/j.websem.2007.11.003. URL: `http://linkinghub.elsevier.com/retrieve/pii/S1570826807000522`.

Ceccaroni, Luigi et al. (Feb. 2009). "PaTac: Urban, Ubiquitous, Personalized Services for Citizens and Tourists". In: *Third International Conference on Digital Society*, pp. 7–12. DOI: 10.1109/ICDS.2009.25. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4782843`.

DailyInfo. Accessed on 18/07/2011. URL: `http://www.dailyinfo.co.uk`.

DateTime ontology specification. Accessed on 13/06/2011. URL: `http://multimedialab.elis.ugent.be/users/gmartens/Ontologies/PecMan/V2.0/DateTime.owl`.

Davies, Stephen, Chris Donaher, and Jesse Hatfield (2010). "Making the Semantic Web usable: interface principles to empower the layperson". In: *Journal of Digital Information* 12.1. URL: `http://journals.tdl.org/jodi/article/view/746/1182`.

DBPedia website. Accessed on 04/08/2011. URL: `http://dbpedia.org`.

Delbru, Renaud (2010). *Comments on Jena LARQ, Sesame Lucene SAIL and SIREn.* Accessed on 16/07/2011. URL: `http://lists.deri.org/pipermail/siren/2010-January/000013.html`.

Ermilov, Timofey et al. (2011). "OntoWiki MobileKnowledge Management in your Pocket". In: *Proceedings of the ESWC2011.* URL: `http://svn.aksw.org/papers/2011/WWW\_Ontowiki\_Mobile/public.pdf`.

Eventful. Accessed on 10/08/2011. URL: `http://eventful.com/oxford/events`.

Facebook Developers Graph API documentation about events (2011). Accessed on 03/06/2011. URL: `https://developers.facebook.com/docs/reference/api/event/`.

Facebook's Statement of Rights and Responsibilites. Accessed on 23/06/2011. URL: `https://www.facebook.com/terms.php`.

Finelli, Thomas, Catherine O'Brien, and Kevin Scannell (2010). *Venipedia.* Tech. rep. Worcester Polytechnic Institute - Venice Project Center. URL: `http://www.venipedia.org`.

Flouch, Hugh and Kevin Harris (2010). *Online Neighbourhood Networks Study*. Tech. rep. November. London: Networked Neighbourhoods. URL: `http://networkedneighbourhoods.com/?page\_id=409`.

Fouad, R.A. et al. (2010). "On Location-Centric Semantic Information Retrieval in Ubiquitous Computing Environments". In: *INTERNATIONAL JOURNAL OF ELECTRICAL & COMPUTER SCIENCES* December. URL: `http://www.ijens.org/106306-9898IJECS-IJENS.pdf`.

Foursquare Venues Project homepage. Accessed on 12/06/2011. URL: `https://developer.foursquare.com/venues/`.

GeoARQ website. Accessed on 12/08/2011. URL: `https://github.com/castagna/GeoARQ`.

GeoNames. Accessed on 05/07/2011. URL: `http://www.geonames.org/`.

GeoNames ontology specification. Accessed on 16/07/2011. URL: `http://www.geonames.org/ontology/documentation.html`.

GoodRelations homepage. Accessed on 20/07/2011. URL: `http://www.heppnetz.de/projects/goodrelations/`.

GRDDL. Accessed on 12/06/2011. URL: `http://www.w3.org/2004/01/rdxh/spec`.

Gruber, T (Dec. 2007). "Collective knowledge systems: Where the Social Web meets the Semantic Web". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 6, pp. 4–13. ISSN: 15708268. DOI: `10.1016/j.websem.2007.11.011`. URL: `http://linkinghub.elsevier.com/retrieve/pii/S1570826807000583`.

iCal4j website. Accessed on 18/07/2011. URL: `http://www.ical4j.org`.

Ioannou, Ekaterini et al. (2010). "Efficient Semantic-Aware Detection of Near Duplicate Resources". In: *The Semantic Web: Research and Applications*. Lecture Notes in Computer Science 6089. Ed. by Lora Aroyo et al., pp. 136–150. DOI: `10.1007/978-3-642-13489-0\_10`. URL: `http://www.springerlink.com/index/6066160G04877230.pdf`.

Janowicz, K. (2010). "The role of space and time for knowledge organization on the semantic web". In: *Semantic Web-Interoperability, Usability, Applicability (2010* 1, pp. 25–32. DOI: `10.3233/SW-2010-0001`. URL: `http://scholar.google.com/scholar?hl=en\&btnG=Search\&q=intitle:The+role+of+space+and+time+for+knowledge+organization+on+the+Semantic+Web\#0`.

JChronic website. Accessed on 15/08/2011. URL: `https://github.com/samtingleff/jchronic`.

Jersey website. Accessed on 27/07/2011. URL: `http://jersey.dev.java.net`.

JSoup website. Accessed on 18/08/2011. URL: `http://jsoup.org`.

Kuhn, Tobias (2008). "Acewiki: Collaborative ontology management in controlled natural language". In: *Proceedings of the 3rd Semantic Wiki Workshop*. Vol. 360. arXiv:`arXiv:0807.4623v1`. URL: `http://arxiv.org/pdf/0807.4623`.

Langegger, Andreas, Wolfram Wöß, and Martin Blöchl (2008). "A Semantic Web Middleware for Virtual Data Integration on the Web". In: *In Bechhofer S Hauswirth M Hoffmann J Koubarakis M*. Lecture Notes in Computer Science vol. Ed. by Sean Bechhofer et al., pp. 493–507. ISSN: 03029743. DOI: `10.1007/978-3-540-68234-9\_37`. URL: `http://www.springerlink.com/index/1804822517684043.pdf`.

Lanyrd website. Accessed on 18/07/2011. URL: `http://lanyrd.com/`.

LARQ website. Accessed on 20/08/2011. URL: `http://jena.sourceforge.net/ARQ/lucene-arq.html`.

Layar website. Accessed on 18/08/2011. URL: `http://www.layar.com/`.

Le-Phuoc, Danh et al. (2009). "Rapid prototyping of semantic mash-ups through semantic web pipes". In: *Proceedings of the 18th international conference on World wide web - WWW '09*, p. 581. DOI: `10.1145/1526709.1526788`. URL: `http://portal.acm.org/citation.cfm?doid=1526709.1526788`.

LinkedGeoData. Accessed on 21/05/2011. URL: `http://linkedgeodata.org/`.

LODE: an ontology for Linking Open Descriptions of Events specification. Accessed on 26/06/2011. URL: `http://linkedevents.org/ontology/`.

Matyas, Sebastian et al. (2008). "Designing location-based mobile games with a purpose: collecting geospatial data with CityExplorer". In: *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pp. 244–247. URL: `http://portal.acm.org/citation.cfm?id=1501806`.

Meilender, Thomas et al. (2011). "Les moteurs de wikis sémantiques : un état de lart". In: *Extraction et gestion des connaissances (EGC 2011)*. Brest, France. URL: `http://hal.archives-ouvertes.fr/hal-00542813/`.

Nachouki, Gilles and Mohamed Quafafou (Apr. 2011). "MashUp web data sources and services based on semantic queries". In: *Information Systems* 36.2, pp. 151–173. ISSN: 03064379. DOI: `10.1016/j.is.2010.08.001`. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0306437910000785`.

Omitola, Tope et al. (2010). "Put in your postcode, out comes the data: A Case Study". In: *The Semantic Web: Research and Applications*, pp. 318–332. URL: `http://www.springerlink.com/index/F83H541WG8783578.pdf`.

OntoWiki demonstration. Accessed on 12/08/2011. URL: http://ontowiki.net/Projects/OntoWiki.

Opening Times website. Accessed on 18/07/2011. URL: http://opening-times.co.uk/.

OpenStreetMap. Accessed on 21/07/2011. URL: http://www.openstreetmap.org/.

OWL-Time specification. Accessed on 16/07/2011. URL: http://www.w3.org/TR/owl-time/.

Oxford City Council's website local view. Accessed on 12/08/2011. URL: http://www.oxford.gov.uk/PageRender/decVanilla/LocalView.htm.

Oxford Fairtrade Hack-day. Accessed on 14/07/2011. URL: http://oxfairtrade.wordpress.com/2011/06/15/oxford-fairtrade-hack-day-create-something-that-makes-a-difference/.

OxPoints. Accessed on 20/07/2011. URL: http://www.oucs.ox.ac.uk/oxpoints/.

Palmonari, Matteo et al. (Apr. 2011). "Aggregated search of data and services". In: *Information Systems* 36.2, pp. 134–150. ISSN: 03064379. DOI: 10.1016/j.is.2010.09.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S0306437910000979.

Passant, Alexandre and P.N. Mendes (2010). "sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub". In: *Scripting for the Semantic Web Workshop (SFSW2010) at ESWC2010*, pp. 1–10. URL: http://www.semanticscripting.org/SFSW2010/papers/sfsw2010_submission_6.pdf.

Peng, Zhipeng et al. (2010). "Semantic-based Mobile Mashup Platform". In: *Proceedings of the 9th International Semantic Web Conference ISWC 2010.* Ed. by Peter F Patel-Schneider et al. LNCS. Springer, pp. 1–4.

PingTheSemanticWeb statistics on namespaces (2011). Accessed on 30/06/2011. URL: http://pingthesemanticweb.com/stats/namespaces.php.

ProductDB website.

Protege website. Accessed on 23/07/2011. URL: http://protege.stanford.edu/.

PubSubHubbub project. Accessed on 20/06/2011. URL: http://code.google.com/p/pubsubhubbub/.

Representing vCard Objects in RDF (2010). Accessed on 13/06/2011. URL: http://www.w3.org/Submission/vcard-rdf/.

Reynolds, Vinny et al. (2010). "Exploiting Linked Open Data for Mobile Augmented Reality". In: *W3C Workshop: Augmented Reality on the Web (June 2010).* URL: http://www.w3.org/2010/06/w3car/exploiting_lod_for_ar.pdf.

Seevl. Accessed on 25/08/2011. URL: http://www.seevl.net.

Seevl An interview with Alexandre Passant. Accessed on 25/08/2011. URL: http://semanticweb.com/seevl-part-i-?-an-interview-with-alexandre-passant_b21964.

Segaran, T. and J. Hammerbacher (2009). "Data Finds Data". In: *Beautiful data: the stories behind elegant data solutions*. O'Reilly Media. Chap. 9. ISBN: 0596157118.

Semantic MediaWiki. Accessed on 14/08/2011. URL: http://semantic-mediawiki.org.

SPARQL Endpoint definition. Accessed on 27/08/2011. URL: http://semanticweb.org/wiki/SPARQL_endpoint.

Temporal Ontology specification. Accessed on 20/06/2011. URL: {http://www.service-description.com/NFP/owl/single/Temporal.owl\}.

The Event Ontology specification. Accessed on 12/06/2011. URL: http://motools.sourceforge.net/event/event.html.

Torres, Diego et al. (2011). "Semdrops: A Social Semantic Tagging Approach for Emerging Semantic Data". In: *2011 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2011)*. Wi. URL: http://hal.inria.fr/inria-00599031/.

Tramp, S., P. Frischmuth, and N. Heino (2010). "OntoWiki - A Semantic Data Wiki Enabling the Collaborative Creation and (Linked Data) Publication of RDF Knowledge Bases". In: *Demo Proceedings of the EKAW 2010*. Springer. URL: http://svn.aksw.org/papers/2010/EKAW_Demo_OntoWiki/public.pdf.

University of Oxford - Oxford talks website. Accessed on 14/08/2011. URL: http://talks.linacre.ox.ac.uk/.

Valle, Emanuele Della, Irene Celino, and Daniele Dell'Aglio (Apr. 2010). "The Experience of Realizing a Semantic Web Urban Computing Application". In: *Transactions in GIS* 14.2, pp. 163–181. ISSN: 13611682. DOI: 10.1111/j.1467-9671.2010.01189.x. URL: http://doi.wiley.com/10.1111/j.1467-9671.2010.01189.x.

Visit Oxfordshire. URL: http://www.visitoxfordandoxfordshire.com/default.aspx.

Wikipedia on HTTP. Accessed on 27/08/2011. URL: http://en.wikipedia.org/wiki/HTTP.

Wikipedia on Linked Data. Accessed on 27/08/2011. URL: http://en.wikipedia.org/wiki/Linked_Data.

Wikipedia on ontology. Accessed on 18/08/2011. URL: http://en.wikipedia.org/wiki/Ontology_(computer_science).

Wikipedia on OWL. Accessed on 27/08/2011. URL: `http://en.wikipedia.org/wiki/Web\_Ontology\_Language`.

"Wikipedia on RDF". In: Accessed on 24/08/2011. URL: `http://en.wikipedia.org/wiki/Resource\_Description\_Framework`.

Wikipedia on RDFS. Accessed on 27/08/2011. URL: `http://en.wikipedia.org/wiki/RDFS`.

Wikipedia on REST. Accessed on 27/08/2011. URL: `http://en.wikipedia.org/wiki/REST`.

Wikipedia on Urban Computing. Accessed on 18/08/2011. URL: `http://en.wikipedia.org/wiki/Urban\_computing`.

Wikipedia on URI. Accessed on 27/08/2011. URL: `http://en.wikipedia.org/wiki/URI`.

Wikipedia on URL. Accessed on 27/08/2011. URL: `http://en.wikipedia.org/wiki/URL`.

XProc. Accessed on 01/06/2011. URL: `http://www.w3.org/TR/xproc/`.

Bibliography

Alesso, H. Peter and Craig F. Smith (2006). *Thinking on the Web: Berners-Lee, Gödel and Turing*. Wiley-Interscience. ISBN: 9780471768142. URL: http://amazon.com/o/ASIN/0471768146/.

Della Valle, E. et al. (2008). "Urban Computing: a challenging problem for Semantic Technologies". In: *Workshop on New forms of Reasoning for the Semantic Web: scalable, tolerant and dynamic (NEFORS 2008), colocated with the 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand.* URL: http://www.larkc.eu/wp-content/uploads/2008/11/nefors08_submission_2.pdf.

Heath, Tom and Christian Bizer (2011). *Linked Data: Evolving the Web into a Global Data Space (Synthesis Lectures on the Semantic Web: Theory and Technology)*. HTML version available at: http://linkeddatabook.com/book. Morgan & Claypool Publishers. ISBN: 1608454304. URL: http://linkeddatabook.com/editions/1.0/.

Hebeler, John et al. (2009). *Semantic Web Programming*. Wiley. ISBN: 9780470418017. URL: http://amazon.com/o/ASIN/047041801X/.

Huang, Shiu-Li, Sheng-Cheng Lin, and Yung-Chun Chan (July 2011). "Investigating effectiveness and user acceptance of semantic social tagging for knowledge sharing". In: *Information Processing & Management*. ISSN: 03064573. DOI: 10.1016/j.ipm.2011.07.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S0306457311000744.

Segaran, Toby (2007). *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly Media. ISBN: 9780596529321. URL: http://amazon.com/o/ASIN/0596529325/.

Segaran, Toby, Colin Evans, and Jamie Taylor (2009). *Programming the Semantic Web*. O'Reilly Media. ISBN: 9780596153816. URL: http://amazon.com/o/ASIN/0596153813/.

Shepard, Mark, ed. (Mar. 2011). *Sentient City: Ubiquitous Computing, Architecture, and the Future of Urban Space*. The MIT Press. ISBN: 9780262515863. URL: `http://amazon.com/o/ASIN/0262515865/`.

Studer, Rudi, Stephan Grimm, and Andreas Abecker, eds. (2007). *Semantic Web Services: Concepts, Technologies, and Applications*. Springer. ISBN: 9783642089879. URL: `http://amazon.com/o/ASIN/3642089879/`.

September 1, 2011