



Χρήση του Σημασιολογικού Ιστού και τεχνικών προσδιορισμού θέσης για τον εντοπισμό σημείων ενδιαφέροντος – Εφαρμογή με DBpedia και Google Latitude

Μεταπτυχιακή εργασία

Μαρία Κομματά
Βασίλειος Γογγολίδης

Επιβλέπων καθηγητής:
Βασίλειος Λούμος

ΔΠΜΣ «Τεχνοοικονομικά Συστήματα»



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ – Σχολή
Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – Τμήμα Βιομηχανικής
Διοίκησης & Τεχνολογίας

02/2012

Copyright © Maria Kommata, Vasilios Gongolidis, 2012
All rights reserved.

Semantic driven location services in the Web of Data - A case of Google Latitude and DBpedia

Thesis

Maria Kommata
Vasilios Gongolidis

Supervisor: Professor
Vasilios Loumos

MBA “Techno-economical Systems”



NATIONAL TECHNICAL UNIVERSITY OF ATHENS –
School of Electrical and Computer Engineering



UNIVERSITY OF PIRAEUS – Department of Industrial
Management and Technology

The work presented in this thesis could not have been concluded without the constant support and guidance of Professor Vasilios Loumos and Scientific Affiliate Agis Papantoniou of Multimedia Technology Laboratory, National Technical University of Athens. Their ideas and remarks provided us with constant inspiration, making the avocation with this thesis a truly pleasant and intriguing experience.

We also want to thank our families for their support in our effort. We promise to reward them for the time we have deprived them but still, we also hope we can be of an example to our children for a life-long pursuit of knowledge.

Special thanks to Demetrios Georgakopoulos for his precise remarks and interest in our work.

Abstract

The wide use and expansion of Internet has resulted to the production and exchange of huge amount of information as well as the development of computer applications that have transformed the ways people perform business and communicate. But exactly these characteristics seem to be the main problems that the Internet is facing nowadays: huge amount of information, dispersed across systems that are not interconnected, suitable only for human-processing while the real-meaning of this information, the *semantics*, is opaque to the machines which therefore are unable to assist users in the processing of this information.

To resolve these problems the XML (eXtensible Markup Language) was developed which is the dominant data-exchange standard between information systems. Along with XML, XML Schema is used for the definition of the structure of XML documents. XML in conjunction with XML Schema form the basis for the structural and syntactic interoperability across the Internet.

Semantic interoperability is the purpose of the Semantic Web. Semantic Web transforms the World Wide Web into reach semantic forms through the use of relevant technologies (RDF, OWL). The main language for addressing semantic queries in the Semantic Web is SPARQL (Simple Protocol and RDF Query Language).

Along with the evolution of technologies in the semantic web area, a rapid development is observed in technologies concerning electronic maps and the ability to visualize on them available information with spatial dimension. The leader in this field is Google Company.

In the context of this thesis, an application has been developed utilizing SPARQL language with the purpose of extracting data from the Semantic Web, according to a user's preferences and the location of the data relevant to the user's position.

More specifically, the user's position is located with the Google Latitude technology. At the next step, after the user selects the category of information that is interested to locate and specify the maximum radius (distance) of this information, a SPARQL query is executed for the retrieval of the relevant information from DBpedia. DBpedia is a project aiming to extract structured content from the information created as part of the Wikipedia project. This structured information is then made available on the World Wide Web.

At the final step, the returned results are visualized on Google Maps at the website under URL <http://worth2c.info> that was developed in the context of this thesis using PHP and Javascript.

Keywords

Semantic Web, HTML, XML, XML Schema, Namespace, Ontology, Linked Open Data, URI, RDF, RDF Schema, JSON, OWL, Dublin Core, FOAF, SKOS, SPARQL, Public Data Source, Virtuoso, SNORQL, W3C, Google Latitude, Google Badge, Google Maps, Google Maps API, Javascript, PHP.

Περίληψη

Η ευρεία εξάπλωση και χρήση του διαδικτύου έχει ως αποτέλεσμα την διάθεση και διακίνηση μεγάλου όγκου πληροφοριών και την ανάπτυξη εφαρμογών οι οποίες έχουν μεταμορφώσει τον τρόπο με τον οποίο οι άνθρωποι συναλλάσσονται και επικοινωνούν μεταξύ τους. Όμως αυτό τείνει να αποτελέσει και το πρόβλημα του διαδικτύου στην σημερινή του μορφή: μεγάλος όγκος πληροφοριών, κατακερματισμένος κυρίως σε μη-διασυνδεδεμένα συστήματα, που είναι κατάλληλος για επεξεργασία μόνο από τον άνθρωπο, αφού δεν υπάρχει η δυνατότητα σημασιολογικής ερμηνείας από τις μηχανές και επακόλουθα οποιαδήποτε αυτοματοποίηση της επεξεργασίας αυτών των πληροφοριών.

Προς την κατεύθυνση επίλυσης του συγκεκριμένου προβλήματος, αναπτύχθηκε η γλώσσα XML (eXtensible Markup Language), η οποία αποτελεί σήμερα το κυρίαρχο πρότυπο ανταλλαγής δεδομένων μεταξύ πληροφοριακών συστημάτων. Η γλώσσα XML Schema χρησιμοποιείται για τον ορισμό της δομής των XML εγγράφων. Η XML σε συνδυασμό με την XML Schema αποτελούν τη βάση της συντακτικής και δομικής διαλειτουργικότητας (Structural and Syntactic Interoperability) στο Διαδίκτυο.

Την ανάγκη για σημασιολογική διαλειτουργικότητα (Semantic interoperability) έρχεται να καλύψει ο Σημασιολογικός Ιστός, δίνοντας την δυνατότητα “μετασχηματισμού” του παγκόσμιου ιστού σε πλούσιες πλέον σημασιολογικές φόρμες με τη χρήση σημασιολογικών τεχνολογιών (RDF, OWL). Η επικρατέστερη γλώσσα σημασιολογικών ερωτήσεων στο περιβάλλον αυτό είναι η γλώσσα SPARQL (Simple Protocol and RDF Query Language).

Ταυτόχρονα, με την ανάπτυξη τεχνολογιών που ασχολούνται με τον σημασιολογικό ιστό, παρατηρείται μια αλματώδης ανάπτυξη των τεχνολογιών που σχετίζονται με την δημιουργία ηλεκτρονικών χαρτών, προκειμένου να προσδιοριστεί η επακριβής θέση διαθέσιμων πληροφοριών που έχουν χωρική διάσταση. Κυρίαρχη θέση στον συγκεκριμένο χώρο έχει αποκτήσει η εταιρεία Google.

Στο πλαίσιο της παρούσας διπλωματικής, αναπτύχθηκε μία εφαρμογή με σκοπό την αξιοποίηση της γλώσσας SPARQL για την άντληση δεδομένων από τον σημασιολογικό ιστό, σύμφωνα με τις προτιμήσεις του χρήστη καθώς και την χωροθέτηση των δεδομένων αυτών σε σχέση με την θέση του χρήστη.

Πιο συγκεκριμένα, αξιοποιώντας την τεχνολογία Google Latitude, προσδιορίζεται η θέση του χρήστη. Στην συνέχεια, αφού ο χρήστης επιλέξει την κατηγορία των πληροφοριών που ενδιαφέρεται να εντοπίσει γύρω του, σε μια ακτίνα που θα καθορίσει, εκτελείται ένα ερώτημα σε SPARQL για την άντληση των επιθυμητών πληροφοριών από την DBpedia. Η DBpedia αποτελεί μια προσπάθεια για την

εξαγωγή, διασύνδεση και επαναχρησιμοποίηση δομημένης πληροφορίας διαμέσου του Web από την εγκυκλοπαίδεια Wikipedia. Τα αποτελέσματα της αναζήτησης απεικονίζονται με την βοήθεια των Google Maps στην δικτυακή θέση <http://worth2c.info> που αναπτύχθηκε για τους σκοπούς της παρούσας διπλωματικής με χρήση τεχνολογιών PHP και Javascript.

Λέξεις – κλειδιά

Σημασιολογικός Ιστός, HTML, XML, XML Schema, Namespace, οντολογία, Linked Open Data, URI, RDF, RDF Schema, JSON, OWL, Dublin Core, FOAF, SKOS, SPARQL, Public Data Source, Virtuoso, SNORQL, W3C, Google Latitude, Google Badge, Google Maps, Google Maps API, Javascript, PHP.

Table of Contents

Abstract	5
Keywords	6
Περίληψη.....	7
Λέξεις – κλειδιά	8
Table of Contents	9
List of Figures	12
1. Introduction.....	15
2. Today's web.....	19
2.1. HTML	19
2.2. XML.....	23
2.2.1. XML concepts	25
2.2.2. XML Schema	27
2.2.3. Namespaces.....	28
2.2.4. XML Formatting	29
2.3. Problems of today's web	30
3. Semantic Web and Linked Open Data	33
3.1. Semantic Web.....	33
3.1.1. Architecture.....	35
3.1.2. Layered approach	37
3.1.3. Knowledge representation.....	38
3.1.4. Ontologies	41
3.2. Linked Open Data	43
4. Technologies and standards of the Semantic Web	49
4.1. Metadata	49
4.2. URLs, URIs, URNs, IRIs and Namespaces	51
4.3. Resource Description Framework (RDF).....	54
4.3.1. Introduction.....	54
4.3.2. Basic Ideas of RDF	55
4.3.3. Triple view of RDF statements.....	56
4.3.3.1. RDF: XML-Based syntax.....	58
4.3.3.2. RDF versus XML	60
4.3.3.3. RDF Serialization Formats	61
4.3.3.3.1. N-Triples.....	61
4.3.3.3.2. Notation 3 (N3) syntax.....	62

4.3.3.3.3.	RDF/XML	64
4.3.3.3.4.	Turtle (Terse RDF Triple Language)	65
4.3.3.3.5.	JSON (JavaScript Object Notation)	67
4.4.	RDF Schema	71
4.4.1.	Classes	71
4.4.2.	Class Hierarchies and Inheritance.....	71
4.5.	Web Ontology Language (OWL).....	74
4.5.1.	Requirements for Ontology Languages.....	74
4.5.2.	Limitations of RDF	75
4.5.3.	The three versions of OWL.....	76
4.5.4.	Brief description of the language.....	78
4.6.	Some well-known initiatives.....	82
4.6.1.	Dublin Core Metadata Initiative (DCMI).....	82
4.6.2.	The Friend of a Friend (FOAF) project.....	83
4.6.3.	DBpedia	86
4.6.4.	Simple Knowledge Organization System (SKOS).....	90
4.7.	SPARQL	93
4.7.1.	SPARQL query basic form	95
4.7.2.	Querying a Public Data Source	97
5.	Roadmap	101
6.	Building our Application - Thesis	103
6.1.	Geocoding.....	104
6.2.	Google Maps.....	104
7.	Step By Step approach	107
7.1.	Step 1. – Where is the user located?.....	107
7.1.1.	Google Latitude	107
7.1.2.	Google Latitude Badge	109
7.1.3.	PHP.....	110
7.1.3.1.	About	110
7.1.3.2.	Syntax.....	110
7.1.4.	JavaScript.....	111
7.1.4.1.	About	111
7.1.4.2.	Features	112
7.1.5.	“Show me the Code!”	115

7.2.	Step 2. – Searching the Semantic Web.....	118
7.2.1.	Inside DBpedia – Specific Vocabulary Elements	118
7.2.2.	Distance between coordinates	120
7.2.3.	“Show me the Code!”	121
7.2.3.1.	Virtuoso – A DBpedia endpoint	121
7.2.3.2.	Deep into SPARQL.....	123
7.2.3.3.	Transform SPARQL results into JSON	130
7.3.	Step 3. – Visualization of the results on the Map	132
7.3.1.	Google Maps API V3.....	132
7.3.1.1.	Connect with the Google Maps API.....	132
7.3.1.2.	Create the Google Map.....	133
8.	Putting it all together	138
8.1.	Domain Name	138
8.2.	Web hosting plan	138
8.3.	Visualization.....	140
9.	Synopsis - Conclusions	141
10.	APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia	147
11.	APPENDIX B – The code	187
12.	Bibliography-References.....	193

List of Figures

Figure 1. Example of a user's SPARQL query against DBpedia to retrieve nearby places of interest	17
Figure 2. Example of simple reasoning	31
Figure 3. Versions of Semantic Web architecture (source: [14])	36
Figure 4. Semantic Web knowledge management architecture	40
Figure 5. Relation of Data	43
Figure 6. Linked Open Data dataset - September 2011 (source: [19])	45
Figure 7. The relation between URI, URL and URN	52
Figure 8. RDF evolution	54
Figure 9. Graphical representation of an RDF Triple	56
Figure 10. Example of an RDF Triple	56
Figure 11. Example of combined graphs	57
Figure 12. An example of a FOAF graph	64
Figure 13. JSON object representation	67
Figure 14. JSON array representation	67
Figure 15. JSON value representation	68
Figure 16. JSON string representation	68
Figure 17. JSON number representation	68
Figure 18. RDF Schema - A hierarchy of classes	72
Figure 19. Evolution from RDF to OWL (source [26])	74
Figure 20. Key purpose of major ontology languages	76
Figure 21. The 3 versions of OWL	77
Figure 22. Example of a Wikipedia Infobox	87
Figure 23. Example of a Wikipedia results page	87
Figure 24. Corresponding DBpedia page of a Wikipedia resource	88
Figure 25. DBpedia Live-System Architecture (source: [32])	89
Figure 26. DBpedia Relationship Finder example	90
Figure 27. Example of RDF graph using the SKOS Core Vocabulary (source: [37])	92
Figure 28. SPARQL query basic form	95
Figure 29. Execution of a SPARQL query	96
Figure 30. SPARQL design: WHERE specifies data to pull out; SELECT picks which data to display (source: [4])	97
Figure 31. SNORQL endpoint of DBpedia	98
Figure 32. SNORQL results page of a sample SPARQL query	99
Figure 33. Google Latitude initial page	108
Figure 34. Application's data entry screenshot	109
Figure 35. DBpedia's sample abstract property	118
Figure 36. DBpedia's sample label property	119
Figure 37. DBpedia's sample subject property	119
Figure 38. Geo-coordinates of Acropolis Museum	120
Figure 39. Virtuoso - A DBpedia SPARQL endpoint	121
Figure 40. Combination of RDF triples	126
Figure 41. Worth2c.info domain name registration	138
Figure 42. Domain worth2c.info WHOIS registration details	138

Semantic Web and Linked Open Data

Figure 43. Purchase of web hosting plan	139
Figure 44. Provider's control panel.....	139
Figure 45. worth2c.info sample page	140
Figure 46. 3-level structure of resource Greece.....	148

1. Introduction

Most of today's Web content is suitable for human consumption. Even Web content that is generated automatically from databases is usually presented without the original structural information found in databases. Typical uses of the Web today involve people's seeking and making use of information, searching for and getting in touch with other people, reviewing catalogs of online stores and ordering products by filling out forms, and viewing material.

These activities are not particularly well supported by software tools. Apart from the existence of links that establish connections between documents, the main valuable, indeed indispensable, tools are search engines. [1]

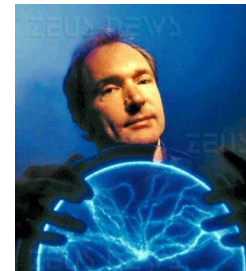
The main idea behind the Semantic Web is the extension of the World Wide Web in a way that information from various, different sources can be easily combined, without depending on applications and websites. Moreover the content, the "semantics" behind the information has to be understood not only by humans, but also by machines. [2]



This idea was Tim Berners-Lee's dream (Semantic Web's inventor), which was described as a machine-to-machine communication, as follows:

*"I have a dream for the Web [in which computers] become capable of **analyzing all the data on the Web** – the content, links and transactions between people and computers. ...the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by **machines talking to machines.**"*

Tim Berners-Lee (1999) Weaving the Web



The Semantic Web is a Web 3.0 technology - a way of linking data between systems or entities that allows for rich, self-describing interrelations of data available across the globe on the web. It involves taking all that information published in HTML documents in different places, and allowing the description of models of data that allow it all to be treated - and researched - as if it were one database. The benefits to the automated research of all the data humanity has to offer on the internet in comparison to today's tools and software are tremendous.

The above mentioned benefits are multiplied when the information is presented to a user relevant to his position as this is defined by a certain latitude and longitude. By

this way, information that is unreachable under certain criteria posed by the user, will be discarded and only truly useful, accessible information will be presented to him. This results in reduced time that is consumed in information retrieval.

Let us assume for example that you are on a holiday trip visiting a place with rich historic past, which is a common scenario in Greece. Unfortunately, you forgot to take a guide with you that will point the exact places of interest. Moreover, you don't have time to read a guide! Under these circumstances, as you drive or walk along, there is a great possibility of missing some really interesting places. But what if you had the chance to use an Internet-capable device and with few simple commands view yourself and all the interesting places around you on a map along with relevant information?

The scope of the current thesis is exactly this: to use the Semantic Web technologies to retrieve information regarding worth to see places that are in proximity to a user's position. For this purpose the following technologies will be used:

- **DBpedia.** DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows a user to ask sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. [3]
- **SPARQL.** SPARQL is a query language (pronounced “sparkle”) designed to pull data from a growing collection of public and private data. Whether this data is part of a semantic web project or an integration of two inventory databases on different platforms behind the same firewall, SPARQL is making it easier to access it. In the words of W3C Director and Web inventor Tim Berners-Lee, “Trying to use the Semantic Web without SPARQL is like trying to use a relational database without SQL.” [4]
- **Google Latitude.** A user's position is provided by Google Latitude. Google Latitude is a location-aware mobile application developed by Google. Latitude lets a mobile phone user to allow certain people to view their current location. Via their own Google Account, the user's cell phone location is mapped on Google Maps. The user can control the accuracy and details of what each of the other users can see — an exact location can be allowed, or it can be limited to identifying the city only. For privacy, it can also be turned off by the user, or a location can be manually entered. Users have to explicitly opt in to Latitude, and may only see the location of those friends who have decided to share their location with them. [5]

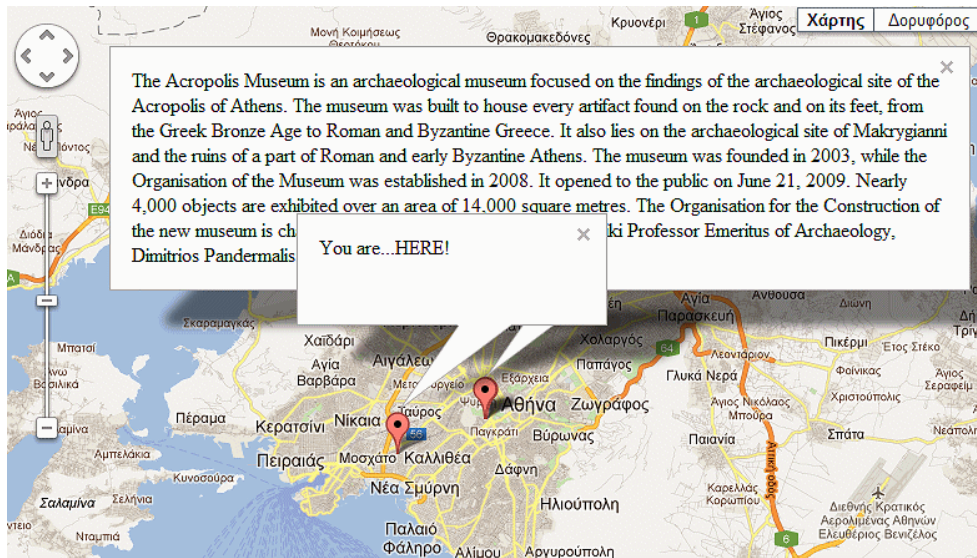


Figure 1. Example of a user's SPARQL query against DBpedia to retrieve nearby places of interest

- **Javascript.** JavaScript is a scripting language you can use — in conjunction with HTML — to create interactive Web pages. JavaScript runs on a user's computer and doesn't require constant downloads from a website.
- **PHP.** A widely-used general-purpose scripting language that is especially suited for Web development, can be embedded into HTML and produce dynamic web pages.

With the above mentioned technologies the user will be able to identify his position using Google Latitude and then query DBpedia with a SPARQL query to retrieve from Wikipedia information that correspond to certain criteria, for example Greek Culture, and is also located in a close distance as defined by the user and measured against the user's position. An example of the integration of these technologies is presented in Figure 1. Example of a user's SPARQL query against DBpedia to retrieve nearby places of interest.

The result of the combination of these technologies will be presented on a website that has been created for the specific purpose. The URL of the site is <http://www.worth2c.info>

2. Today's web

2.1. HTML

The World Wide Web has changed the way people communicate with each other and the way business is conducted. It lies at the heart of a revolution that is currently transforming the developed world toward a knowledge economy and, more broadly speaking, to a knowledge society.

This development has also changed the way people think of computers. Originally they were used for computing numerical calculations. Currently their predominant use is for information processing, typical applications being database systems, text processing, and games. At present there is a transition of focus toward the view of computers as entry points to the information highways. [1]

In the following paragraphs some of the main activities of the current World Wide Web will be presented along with their shortcomings in the use of technology.

Business-to-consumer (B2C) electronic commerce

Business-to-consumer (B2C) electronic commerce is the predominant commercial experience of Web users. A typical scenario involves a user's visiting one or several online shops, browsing their offers, selecting and ordering products.

Ideally, a user would collect information about prices, terms, and conditions (such as availability) of all, or at least all major, online shops and then proceed to select the best offer. But manual browsing is too time-consuming to be conducted on this scale. Typically a user will visit one or a very few online stores before making a decision.

To alleviate this situation, tools for shopping around on the Web are available in the form of shopbots, software agents that visit several shops, extract product and price information, and compile a market overview. Their functionality is provided by wrappers, programs that extract information from an online store. One wrapper per store must be developed. This approach suffers from several drawbacks.

The information is extracted from the online store site through keyword search and other means of textual analysis. This process makes use of assumptions about the proximity of certain pieces of information (for example, the price is indicated by the word *price* followed by the symbol \$ followed by a positive number). This heuristic approach is error-prone; it is not always guaranteed to work. Because of these difficulties only limited information is extracted. For example, shipping expenses,

Semantic Web and Linked Open Data

delivery times, restrictions on the destination country, level of security, and privacy policies are typically not extracted. But all these factors may be significant for the user's decision making. In addition, programming wrappers is time-consuming, and changes in the online store outfit require costly reprogramming. [1]

Business-to-business (B2C) electronic commerce

The greatest economic promise of all online technologies lies in the area of business-to-business (B2B) e-commerce. Traditionally businesses have exchanged their data using the Electronic Data Interchange (EDI) approach. However this technology is complicated and understood only by experts. It is difficult to program and maintain, and it is error-prone. Each B2B communication requires separate programming, so such communications are costly. Finally, EDI is an isolated technology. The interchanged data cannot be easily integrated with other business applications.

The Internet appears to be an ideal infrastructure for business-to-business communication. Businesses have increasingly been looking at Internet-based solutions, and new business models such as *B2B portals* have emerged. Still, B2B e-commerce is hampered by the lack of standards. HTML (hypertext markup language) is too weak to support the outlined activities effectively: it provides neither the structure nor the semantics of information.

Search engines

Keyword-based search engines such as Yahoo and Google are the main tools for using today's Web. It is clear that the Web would not have become the huge success it is, were it not for search engines. However, there are serious problems associated with their use:

- High recall, low precision. Even if the main relevant pages are retrieved, they are of little use if some other million mildly relevant or irrelevant documents are also retrieved. Too much can easily become as bad as too little.
- Low or no recall. Often it happens that we don't get any relevant answer for our request, or that important and relevant pages are not retrieved. Although low recall is a less frequent problem with current search engines, it does occur.
- Results are highly sensitive to vocabulary. Often our initial keywords do not get the results we want; in these cases the relevant documents use different terminology from the original query. This is unsatisfactory because semantically similar queries should return similar results.
- Results are single Web pages. If we need information that is spread over various documents, we must initiate several queries to collect the relevant documents, and then we must manually extract the partial information and put it together.

Interestingly, despite improvements in search engine technology, the difficulties remain essentially the same. It seems that the amount of Web content outpaces technological progress.

But even if a search is successful, it is the person who must browse selected documents to extract the information he is looking for. That is, there is not much support for retrieving the information, a very time-consuming activity. Therefore, the term *information retrieval*, used in association with search engines, is somewhat misleading; *location finder* might be a more appropriate term. Also, results of Web searches are not readily accessible by other software tools; search engines are often isolated applications. [1]

The current web consists of HTML documents constructed from and it depends on visual representation of information through HTML tags. To the extent that computers make sense of the web content it's almost always for the purpose of creating a presentation for the end-user. The real content, the knowledge the files are conveying to the human, is opaque to the computer. Some form of extraction is required to strip off the information part from the presentation part. Other techniques are used to infer meaning from this information and this leads to an increased complexity in the computers dealing with the World Wide Web.

Today Web largely involves human interactivity; the Web pages are mainly built for human consumption and the current technology (HTML language) is concerned with how to present information. Issues, like the actual content and structure of the information or the meaning of information cannot be handled by current web servers. Millions of resources such as HTML files, documents, images and graphics, media files contain a huge amount of scattered information. The retrieval process of the information is hard enough as web is only understood by humans, who alone cannot deal with this huge amount of resources. Even fairly structured information, such as databases, when exported to the Web lose their structure, which should be re-discovered by intelligent text-processing agents, called wrappers.

The main obstacle to providing better support to Web users is that, at present, the meaning of Web content is not *machine-accessible*. Of course, there are tools that can retrieve texts, split them into parts, check the spelling, count their words. But when it comes to *interpreting* sentences and extracting useful information for users, the capabilities of current software are still very limited.

Another problem with the current web is the fact that different terms are used to represent the same meaning, for example in a shopping site could refer to the

shopping cart as cart, while another would refer to it as shopping basket or basket for short, yet another site could refer to it as shopping bag. All these words refer to the same meaning or the same semantics, which is very obvious to humans while it is unknown to computers. These machines have to be explicitly informed that the previous terms are all the same. Another example comes from the fact that the web is multi lingual; so the same term in English is recognized by the computer as a different term if it is written in Greek for example. [2]

Using text processing, how can the current situation be improved? One solution is to use the content as it is represented today and to develop increasingly sophisticated techniques based on artificial intelligence and computational linguistics. This approach has been followed for some time now, but despite some advances the task still appears too ambitious.

An alternative approach is to represent Web content in a form that is more easily machine-processable and to use intelligent techniques to take advantage of these representations. We refer to this plan of revolutionizing the Web as the *Semantic Web* initiative. It is important to understand that the Semantic Web will not be a new global information highway parallel to the existing World Wide Web; instead it will gradually evolve out of the existing Web.

The Semantic Web is propagated by the World Wide Web Consortium (W3C), an international standardization body for the Web. The driving force of the Semantic Web initiative is Tim Berners-Lee, the very person who invented the WWW in the late 1980s. He expects from this initiative the realization of his original vision of the Web, a vision where the meaning of information played a far more important role than it does in today's Web. [1]

The development of the Semantic Web has a lot of industry momentum, and governments are investing heavily. The U.S. government has established the DARPA Agent Markup Language (DAML) Project, and the Semantic Web is among the key action lines of the European Union's Framework Programmes. [6]

2.2. XML

SGML (standard generalized markup language), is an international standard (ISO 8879) for the definition of device and system-independent methods of representing information, both human and machine-readable. Such standards are important because they enable effective communication, thus supporting technological progress and business collaboration. In the WWW area, standards are set by the W3C (World Wide Web Consortium); they are called *recommendations*, in acknowledgment of the fact that in a distributed environment without central authority, standards cannot be enforced. Languages conforming to SGML are called SGML applications.

HTML is a markup language that was derived from SGML because SGML was considered far too complex for Internet-related purposes.

XML (extensible markup language) is another SGML application, and its development was driven by the shortcomings of HTML that were presented in section 2.1 HTML. Some of the motivations for XML can be easily worked out by considering the following example, a Web page that contains information about a particular book.

```
<h2>How To Think Like A Programmer: Problem-solving for the  
Bewildered </h2>  
<i>by <b>Paul Vickers </b></i><br>  
Thomson Learning 2008 <br>  
ISBN 1844809005
```

A typical XML representation of the same information might look like this:

```
<book>  
<title>  
How To Think Like A Programmer: Problem-solving for the  
Bewildered  
</title>  
<author> Paul Vickers </author>  
<publisher> Thomson Learning </publisher>  
<year>2008</year>  
<ISBN>1844809005</ISBN>  
</book>
```

Before we turn to differences between the HTML and XML representations, let us observe a few similarities. First, both representations use *tags*, such as `<h2>` and `</year>`. Indeed both HTML and XML are *markup languages*: they allow one to write some content and provide information about what role that content plays.

Like HTML, XML is based on tags. These tags may be nested (tags within tags). All tags in XML must be closed (for example, for an opening tag `<title>` there must be a closing tag `</title>`), whereas in HTML some tags, such as `
`, may be left open. The enclosed content, together with its opening and closing tags, is referred to as an *element*. (The recent development of XHTML has brought HTML more in line with XML: any valid XHTML document is also a valid XML document, and as a consequence, opening and closing tags in XHTML are balanced.)

A less formal observation is that human users can read both HTML and XML representations quite easily. Both languages were designed to be easily understandable and usable by humans. But how about machines? Imagine an intelligent agent trying to retrieve the names of the author of the book in the previous example. Suppose the HTML page could be located with a Web search (something that is not at all clear; the limitations of current search engines are well documented). There is no *explicit* information as to who the author is. A reasonable guess would be that the author's name appears immediately after the title or immediately follow the word *by*. But there is no guarantee that these conventions are always followed. Clearly, more text processing is needed to answer this question, processing that is open to errors.

The problems arise from the fact that the HTML document does not contain *structural information*, that is, information about pieces of the document and their relationships. In contrast, the XML document is far more easily accessible to machines because every piece of information is described. Moreover, some forms of their *relations* are also defined through the **nesting structure**. For example, the `<author>` tags appear within the `<book>` tags, so they describe properties of the particular book. A machine processing the XML document would be able to deduce that the author element refers to the enclosing book element, rather than having to infer this fact from proximity considerations, as in HTML. An additional advantage is that XML allows the definition of **constraints on values** (for example, that a year must be a number of four digits, that the number must be less than 3,000). *XML allows the representation of information that is also machine-accessible.* [1]

Of course, we must admit that the HTML representation provides more than the XML representation: the formatting of the document is also described. However, this feature is not strength but a weakness of HTML: it *must* specify the formatting; in fact, the main use of an HTML document is to display information (apart from linking to other documents). On the other hand, *XML separates content from formatting*. The same information can be displayed in different ways without requiring multiple copies

of the same content; moreover, the content may be used for purposes other than display.

HTML representations are intended to display information, so the set of tags is fixed: lists, bold, color, and so on. In XML we may use information in various ways, and it is up to the user to define a vocabulary suitable for the application. Therefore, *XML is a metalanguage for markup: it does not have a fixed set of tags but allows users to define tags of their own.*

Just as people cannot communicate effectively if they don't use a common language, applications on the WWW must agree on common vocabularies if they need to communicate and collaborate. Communities and business sectors are in the process of defining their specialized vocabularies, creating XML applications (or extensions; thus the term *extensible* in the name of XML). Such XML applications have been defined in various domains, for example, mathematics (MathML), bioinformatics (BSML), human resources (HRML), astronomy (AML), news (NewsML), and investment (IRML).

Also, the W3C has defined various languages on top of XML, such as SVG and SMIL. This approach has also been taken for RDF (see also section 4.3 Resource Description Framework (RDF)).

It should be noted that XML can serve as a *uniform data exchange format* between applications. In fact, XML's use as a data exchange format between applications nowadays far outstrips its originally intended use as document markup language. Companies often need to retrieve information from their customers and business partners, and update their corporate databases accordingly. If there is not an agreed common standard like XML, then specialized processing and querying software must be developed for each partner separately, leading to technical overhead; moreover, the software must be updated every time a partner decides to change its own database format. [1]

2.2.1. XML concepts

Because XML is designed to describe data and documents, the W3C XML recommendation, is very strict about a small core of format requirements that make the difference between a text document containing a number of tags and an actual XML document [7]. XML documents that meet W3C XML document formatting recommendations are described as being *well-formed* XML documents. Well-formed XML documents can contain elements, attributes, characters, processors and applications, XML Declarations, Tags, Markup and Content.

Elements:

Element is every logical component of a document which either begins with a start-tag and ends with an end-tag, or consists of an empty-element tag. It can possibly contain other elements which are elements-children. For example:

```
<Professor>Vasilios Loumos</Professor>.
```

If there is no content, then the element is called *empty*. An empty element like

```
<Professor> </Professor>.
```

can be abbreviated as

```
<Professor/>
```

Attributes:

Attribute is called a markup structure that consists of a name-value couple which is in a start-tag or empty element tag. Here is an example of an element with 2 attributes:

```
<Professor name="Vasilios Loumos" email="loumos@cs.ntua.gr"/>
```

Unicode Character:

From the definition of the language, it is known that an XML document is alphanumerical consisting of characters. Almost every Unicode character can appear in an XML document.

Processor and Application:

The processor analyzes the markup part of the paper and grants an application to structured information obtained from this. The requirements language defines what a processor should do and what not. From the perspective of the program, there are no specifications on how to manage the data. The processor is often referred to as an XML parser.

XML Declaration:

The XML declaration specifies that the current document is an XML document, and defines the version and the character encoding used in the particular system (such as UTF-8, UTF-16, and ISO 8859-1). The character encoding is not mandatory, but its specification is considered good practice. Sometimes we also specify whether the document is self-contained, that is, whether it does not refer to external structuring documents. An example of an XML declaration is:

```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
```

Tags:

In XML a Tag is what is written between angled brackets, i.e begins with the character "<" and ends with the character ">". The tags are divided in three categories. Initially

there are the start-tags e.g. `<section>`, then we have end-tags e.g. `</section>` and finally we have the empty-element tags, e.g. `<line-break/>`.

Markup and Content:

The characters that compose an XML document are shared on markup and content. These two should be separated from the application with some simple syntax rules. For example all strings that belong to markup begin or end with "<" and ">". Furthermore, it may start with the character "&" and end with character ";". The string which do not form markup is content.

2.2.2. XML Schema

Imagine two applications that try to communicate, and that they wish to use the same vocabulary based on XML. For this purpose it is necessary to define all the element and attribute names that may be used. Moreover, the structure should also be defined: what values an attribute may take, which elements may or must occur within other elements, and so on. In the presence of such structuring information users have an enhanced possibility of document validation. An XML document is *valid* if it is well-formed, uses structuring information, and respects that structuring information. The current way of defining the structure of XML documents is XML Schema, evolution of the older DTD. [1] [8]

XML Schema offers a significantly rich language for defining the structure of XML documents. The most important characteristics of XML Schema are:

- Its syntax is based on XML itself. This design decision provides a significant improvement in readability, but more important, it also allows significant reuse of technology.
- An important improvement is the possibility of reusing and refining schemas. XML Schema allows one to define new types by extending or restricting already existing ones. In combination with an XML-based syntax, this feature allows one to build schemas from other schemas, thus reducing the workload.
- Finally, XML Schema provides a sophisticated set of data types that can be used in XML documents.

An XML schema is an element with an opening tag like

```
<xsd:schema  
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"  
  version="1.0">
```

The element uses the schema of XML Schema found at the W3C Web site. It is, so to speak, the foundation on which new schemas can be built [9]. The prefix `xsd` denotes

Semantic Web and Linked Open Data

the namespace of that schema. The most important contents of an XML Schema are the definitions of element and attribute types, which are defined using data types. XML Schema provides powerful capabilities for defining data types. First there is a variety of *built-in data types*. Here is a list of few:

- Numerical data types, including *integer*, *short*, *Byte*, *long*, *float*, *decimal*
- String data types, including *string*, *ID*, *IDREF*, *CDATA*, *language*
- Date and time data types, including *time*, *date*, *gMonth*, *gYear*

There are also *user-defined data types*, comprising *simple data types*, which cannot use elements or attributes, and *complex data types*, which can use elements and attributes. Complex types are defined from already existing data types by defining some attributes (if any) and using

- *sequence*, a sequence of existing data type elements, the appearance of which in a predefined order is important,
- *all*, a collection of elements that must appear but the order of which is not important,
- *choice*, a collection of elements, of which one will be chosen.

2.2.3. Namespaces

One of the main advantages of using XML as a universal (meta) markup language is that information from various sources may be accessed; in technical terms, an XML document may use more than one schema (or previously DTD). But since each structuring document was developed independently, *name clashes* appear inevitable. If Schema A and Schema B define an element type *e* in different ways, a parser that tries to validate an XML document in which an *e* element appears must be told which Schema to use for validation purposes.

The technical solution is simple: disambiguation is achieved by using a different prefix for each schema. The prefix is separated from the local name by a colon (prefix:name) and have the form `xmlns:prefix="location"`. For example

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

2.2.4. XML Formatting

Unlike HTML documents, XML documents do not contain formatting information. The advantage is that a given XML document can be presented in various ways, when different *style sheets* are applied to it. Style sheets can be written in various languages, for example, in CSS2 (cascading style sheets level 2), XSL (extensible stylesheet language) or Javascript.

Browsers which read XML will accept and use a CSS stylesheet at a minimum, but also the more powerful XSLT stylesheet language can be used to transform XML into HTML—which browsers, of course, already know how to display (and that HTML can still use a CSS stylesheet). This way you get all the document management benefits of using XML, but you don't have to worry about readers needing XML smarts in their browsers. [10]

As with any system where files can be viewed at random by arbitrary users, the author cannot know what resources (such as fonts) are on the user's system, so the same care is needed as with HTML using fonts. To invoke a stylesheet from an XML file for standalone processing in the browser, include one of the stylesheet declarations:

```
<?xml-stylesheet href="abc.xsl" type="text/xsl"?>  
<?xml-stylesheet href="abc.css" type="text/css"?>
```

Specifications for a non-intrusive mechanism, using a processing instruction, to provide links to one or more style sheets, i.e. resources specifying the desired rendering in a designated language are provided by W3C . User agents can use these resources to control presentation of XML. [11]

2.3. Problems of today's web

As it was mentioned in section 2.1 - HTML, the World Wide Web consists mainly of HTML documents which deal with the formatting and presentation of the information having no knowledge, whatsoever, of the meaning of this information. Information is comprehended by humans but it is almost opaque to computers. The vast amount of information makes human processing very difficult to achieve.

XML on the other hand, is a universal metalanguage for defining markup. It provides a uniform framework and a set of tools like parsers for interchange of data and metadata between applications. However, XML does not provide any means of talking about the *semantics* (meaning) of data. For example, there is no intended meaning associated with the nesting of tags; it is up to each application to interpret the nesting. Let us illustrate this point using an example. Suppose we want to express the following fact:

Vasilios Loumos is a lecturer of Multimedia Technology.

There are various ways of representing this sentence in XML. Three possibilities are:

```
<course name="Multimedia Technology">
  <lecturer> Vasilios Loumos</lecturer>
</course>

<lecturer name="Vasilios Loumos">
  <teaches> Multimedia Technology</teaches>
</lecturer>

<teachingOffering>
  <lecturer> Vasilios Loumos</lecturer>
  <course> Multimedia Technology</course>
</teachingOffering>
```

The first two formalizations include essentially an opposite nesting although they represent the same information. So there is no standard way of assigning meaning to tag nesting.

As a result, in either of today's web technologies-HTML and XML-the meaning of the information is incomprehensible to the machines and therefore no reasoning can be deducted from it. To illustrate this more clearly, let us use the following example. In Figure 2. Example of simple reasoning, a simple for humans hierarchy of the real world is depicted.

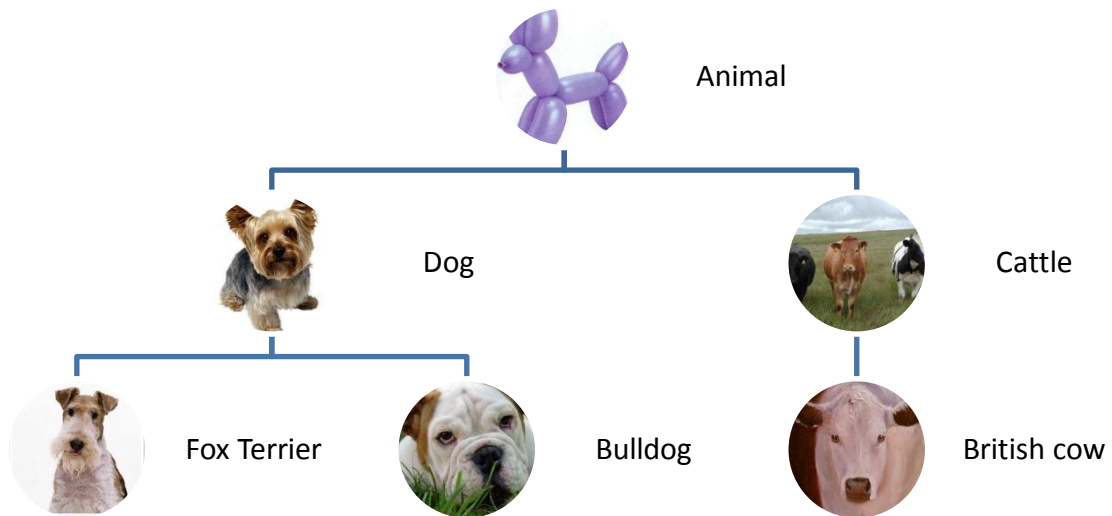


Figure 2. Example of simple reasoning

HTML can make no reasoning whatsoever about this hierarchy and it deals only with the representation of it as a web page.

XML on the other hand, can use various ways to represent this hierarchy, as it was mentioned in the previous example, but cannot make the simple inference that a Fox Terrier dog is an animal!

3. Semantic Web and Linked Open Data

3.1. Semantic Web

In the writing “Through the Looking-Glass”, the sequel of “Alice’s Adventures in Wonderland”, the author Charles Lutwidge Dodgson (pseudonym Lewis Carroll) discusses semantics and pragmatics in a dialogue between Alice and Humpty Dumpty:

“I don’t know what you mean by ‘glory,’ ” Alice said.



Humpty Dumpty smiled contemptuously. “Of course you don’t—till I tell you. I meant ‘there’s a nice knock-down argument for you!’ ”

“But ‘glory’ doesn’t mean ‘a nice knock-down argument’,” Alice objected.

“When I use a word,” Humpty Dumpty said, in rather a scornful tone, “**it means just what I choose it to mean**—neither more nor less.”

“The question is,” said Alice, “whether you can **make words mean so many different things.**”

“The question is,” said Humpty Dumpty, “**which is to be master that’s all.**”

Alice was too much puzzled to say anything, so after a minute Humpty Dumpty began again. “They’ve a temper, some of them—particularly verbs, they’re the proudest—adjectives you can do anything with, but not verbs—however, I can manage the whole lot! Impenetrability! That’s what I say!”

The above dialogue between Alice and Humpty Dumpty, denotes what we already know; there is a difficulty among humans to understand words since they don’t have a single meaning, yet eventually it is fairly easy for humans to communicate effectively. Natural language is amazing. Without any effort you can ask a stranger how to find the nearest coffee shop; you can share your knowledge of music with friends; you can go

Semantic Web and Linked Open Data

to the library, pick up a book, and learn from an author who lived hundreds of years ago. As a simple example, think about the following two sentences. Both are of the form “subject-verb-object,” one of the simplest possible grammatical structures:

1. *Maria enjoys mushrooms.*
2. *Mushrooms scare Billy.*

Each of these sentences represents a piece of information. The words “Billy” and “Maria” refer to specific people, the word “mushroom” refers to a class of organisms, and the words “enjoys” and “scare” tell you the relationship between the person and the organism. Because you know from previous experience what the verbs “enjoy” and “scare” mean, and you’ve probably seen a mushroom before, you’re able to understand the two sentences. And now that you’ve read them, you’re equipped with new knowledge of the world. This is an example of semantics: symbols can refer to things or concepts, and sequences of symbols convey meaning. You can now use the meaning that you derived from the two sentences to answer simple questions such as “Who likes mushrooms?”

Semantics is the process of communicating enough meaning to result in an action. A sequence of symbols can be used to communicate meaning, and this communication can then affect behavior. The current Web works well because humans are very flexible data processors. Whether the information on a web page is arranged as a table, an outline, or a multipage narrative, we are able to extract the important information and use it to guide further knowledge discovery. However, this heterogeneity of information is indecipherable to machines, and the wide range of representations for data on the Web only compounds the problem. If the diversity of information available on the Web can be encoded by content providers into semantic data structures, any application could access and use the rich array of data we have come to rely on. In this vision, data is seamlessly woven together from disparate sources, and new knowledge is derived from the confluence. This is the vision of the semantic web. [12]

Like biological organisms, computers operate in complex, interconnected environments where each element of the system constrains the behavior of many others. Similar to predator-prey relationships, applications and the data they consume tend to follow co-evolutionary paths. Cumulative changes in an application eventually require modification to the data structures on which it operates. Conversely, when enhancements to a data source emerge, the structures for expressing the additional information generally force applications to change. Unfortunately, because of the significant efforts involved, this type of lock-step evolution tends to dampen enhancements in both applications and data sources.

At their core, *semantic technologies decouple applications from data* through the use of a simple, abstract model for knowledge representation. This model releases the mutual constraints on applications and data, allowing both to evolve independently. And by design, this degree of application-data independence promotes data portability. Any application that understands the model can consume any data source using the model. It is this level of data portability that underlies the notion of a machine-readable semantic web.

The basic idea is to **give information a well-defined meaning**, thus better enabling agents and people to work in cooperation. Therefore, the main aim of the Semantic Web is to organize the information founded in the Web (mainly in Web pages) in a better fashion and interconnect the various pieces of information so that the very same information can be used for discovery, automatization, aggregation, and reuse from various, different and disparate applications, that have not been designed either to work together or to work with every different piece of information founded in the Web. [2]

"The Semantic Web is about two things. It is about common formats for interchange of data, where on the original Web we only had interchange of documents. Also it is about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing". (<http://www.w3.org/2001/sw/>) [13]

The Semantic Web aims at machine-processable information. The step from the current Web to the Semantic Web is the step from the manual to the automatic processing of information. This step is comparable to the step from the manual processing of goods to the machine processing of goods at the beginning of the industrial revolution. Hence, the Semantic Web can be seen as the dawn of the informational revolution.

3.1.1. Architecture

Since the publication of the original Semantic Web vision of Berners-Lee, Hendler and Lassila proposed four versions of Semantic Web architecture. All of the architecture versions were presented by Berners-Lee in presentations; they were never published in literature or included as part of a W3C Recommendation. The different versions are depicted as V1 to V4 in Figure 3. Versions of Semantic Web architecture. [2]

Semantic Web and Linked Open Data

In support of the founding vision of the Semantic Web, the first version (V1) of a layered architecture was introduced in 2000 by Berners-Lee. As participant in the ongoing activities of the W3C, Berners-Lee proposed a second version (V2) of the Semantic Web reference architecture in 2003 as part of a presentation at the SIIA Summit. V2 was furthermore presented as part of two presentations in 2003. Berners-Lee proposed a third version (V3) at WWW2005, and in his keynote address at AAAI2006 in July 2006, Berners-Lee introduced the latest (V4) version of the Semantic Web reference architecture. [14]

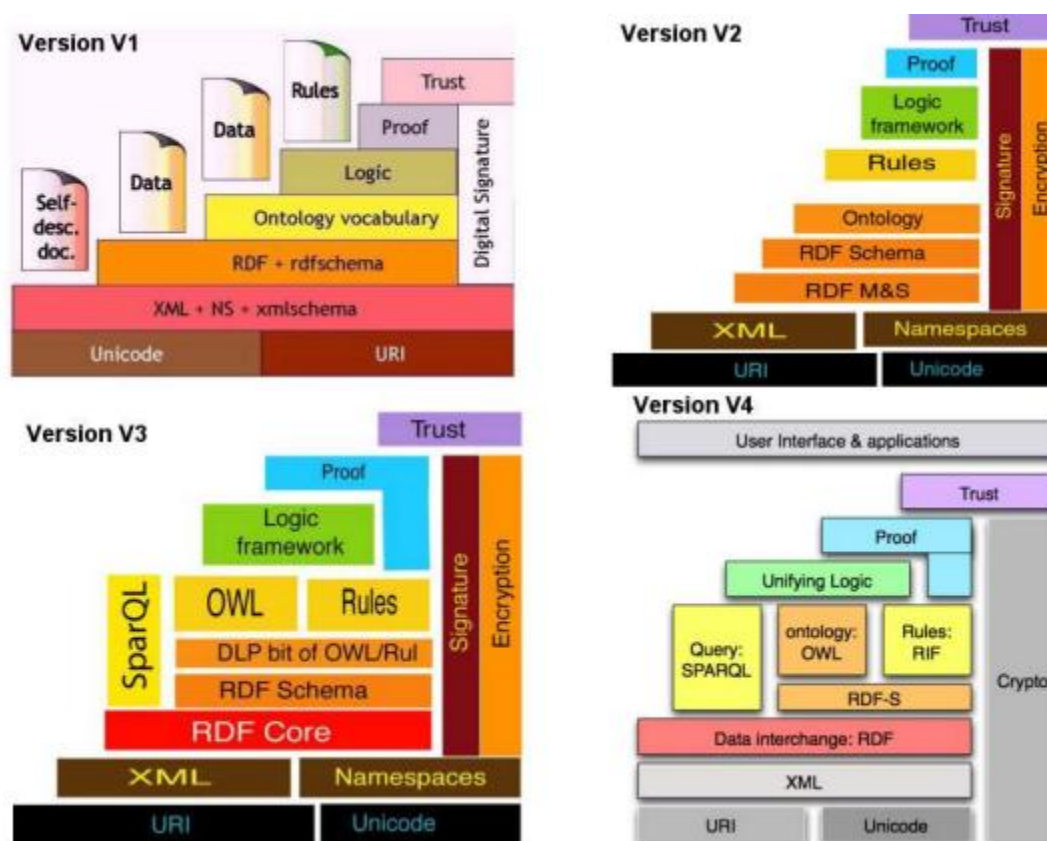


Figure 3. Versions of Semantic Web architecture (source: [14])

Figure 3. Versions of Semantic Web architecture V2-V4 shows alternative layer stacks that takes recent developments into account. The main differences, compared to initial stack, are the following:

- The ontology layer is instantiated with two alternatives: the current standard Web ontology language, OWL, and a rule-based language. Thus an alternative stream in the development of the Semantic Web appears.
- Description Logic Programs (DLP) is the intersection of OWL and Horn logic, and serves as a common foundation.

3.1.2. Layered approach

The development of the Semantic Web proceeds in steps, each step building a *layer* on top of another. The pragmatic justification for this approach is that it is easier to achieve consensus on small steps, whereas it is much harder to get everyone on board if too much is attempted. Usually there are several research groups moving in different directions; this competition of ideas is a major driving force for scientific progress. However, from an engineering perspective there is a need to standardize. So, if most researchers agree on certain issues and disagree on others, it makes sense to fix the points of agreement. This way, even if the more ambitious research efforts should fail, there will be at least partial positive outcomes.

Once a standard has been established, many more groups and companies will adopt it, instead of waiting to see which of the alternative research lines will be successful in the end. The nature of the Semantic Web is such that companies and single users must build tools, add content, and use that content. We cannot wait until the full Semantic Web vision materializes—it may take another ten years for it to be realized to its full extent as envisioned today.

In building one layer of the Semantic Web on top of another, two principles should be followed:

- Downward compatibility. Agents fully aware of a layer should also be able to interpret and use information written at lower levels. For example, agents aware of the semantics of OWL can take full advantage of information written in RDF and RDF Schema.
- Upward partial understanding. The design should be such that agents fully aware of a layer should be able to take at least partial advantage of information at higher levels. For example, an agent aware only of the RDF and RDF Schema semantics might interpret knowledge written in OWL partly, by disregarding those elements that go beyond RDF and RDF Schema. Of course, there is no requirement for all tools to provide this functionality; the point is that this option should be enabled.

While these ideas are theoretically appealing and have been used as guiding principles for the development of the Semantic Web, their realization in practice turned out to be difficult, and some compromises needed to be made.

Figure 3. Versions of Semantic Web architecture shows the “layer cake” of the Semantic Web (due to Tim Berners-Lee), which describes the main layers of the Semantic Web design and vision.

Near the bottom we find *XML*, a language that lets one write structured Web documents with a user-defined vocabulary. XML is particularly suitable for sending documents across the Web.

RDF is a basic data model, like the entity-relationship model, for writing simple statements about Web objects (resources). The RDF data model does not rely on XML, but RDF has an XML-based syntax. Therefore, it is located on top of the XML layer.

RDF Schema provides modeling primitives for organizing Web objects into hierarchies. Key primitives are classes and properties, subclass and subproperty relationships, and domain and range restrictions. RDF Schema is based on RDF. RDF Schema can be viewed as a primitive language for writing ontologies. But there is a need for more powerful *ontology languages* that expand RDF Schema and allow the representations of more complex relationships between Web objects.

The *Logic* layer is used to enhance the ontology language further and to allow the writing of application-specific declarative knowledge.

The *Proof* layer involves the actual deductive process as well as the representation of proofs in Web languages (from lower levels) and proof validation.

Finally, the *Trust* layer will emerge through the use of *digital signatures* and other kinds of knowledge, based on recommendations by trusted agents or on rating and certification agencies and consumer bodies. Sometimes “Web of Trust” is used to indicate that trust will be organized in the same distributed and chaotic way as the WWW itself. Being located at the top of the pyramid, trust is a high-level and crucial concept: the Web will only achieve its full potential when users have trust in its operations (security) and in the quality of information provided.

3.1.3. Knowledge representation

Knowledge representation and *reasoning* aim at designing computer systems that reason about a machine-interpretable representation of the world, similar to human reasoning. *Knowledge-based systems* have a computational model of some domain of interest in which symbols serve as surrogates for real world-domain artefacts, such as physical objects, events, relationships, etc. The domain of interest can cover any part of the real world of any hypothetical system about which one desires to represent knowledge for computational purposes. [15]

A knowledge-based system maintains a knowledge base which stores the symbols of the computational model in form of statements about the domain and it performs

reasoning by manipulating these symbols. Applications can base their decisions on domain-relevant questions posed to a knowledge base.

Semantic Web has a lot in common with Artificial Intelligence as it reuses lots of classic methods of knowledge and reasoning representation, such as semantic networks, frameworks and logic adapted to the open and unpredictable environment of the Internet. With the Semantic Web, the methods of knowledge representation and reasoning in Artificial Intelligence find the most important application in a project of enormous proportions, as is interoperability of data and online applications.

Another important application of Artificial Intelligence in Semantic Web is the intelligent agents, whose technology so far has been either detached from the network, resulting in little diffusion, or was using custom solutions for the interoperability of information on network, resulting in high cost for developing applications.

Using the Semantic Web technologies from websites and search engines, agents will have the information needed and will be independent of the human user. This information will be associated with a shared reference framework (ontology) so that their meaning will be accessible and understandable by all agents acting on network, regardless of the purpose of which they were developed. [2]

Furthermore, an attempt is made to develop technologies that assist the interoperability of data and online applications. So, proposes languages that express information in a manner understandable by the agents and also encourages network users to enrich their documents with such information. Hence comes the definition of the Semantic Web as an information-oriented, meaning, something like a global database and knowledge.

If the ultimate goal of artificial intelligence is to build an intelligent agent exhibiting human-level intelligence (and higher), the goal of the Semantic Web is to assist human users in their day-to-day online activities. It is clear that the Semantic Web will make extensive use of current artificial intelligence technology and that advances in that technology will lead to a better Semantic Web. But there is no need to wait until artificial intelligence reaches a higher level of achievement; current artificial intelligence technology is already sufficient to go a long way toward realizing the Semantic Web vision. [1]

Figure 4. Semantic Web knowledge management architecture shows the architecture for knowledge management based on the Semantic Web. The architecture addresses all the key stages of the knowledge management lifecycle.

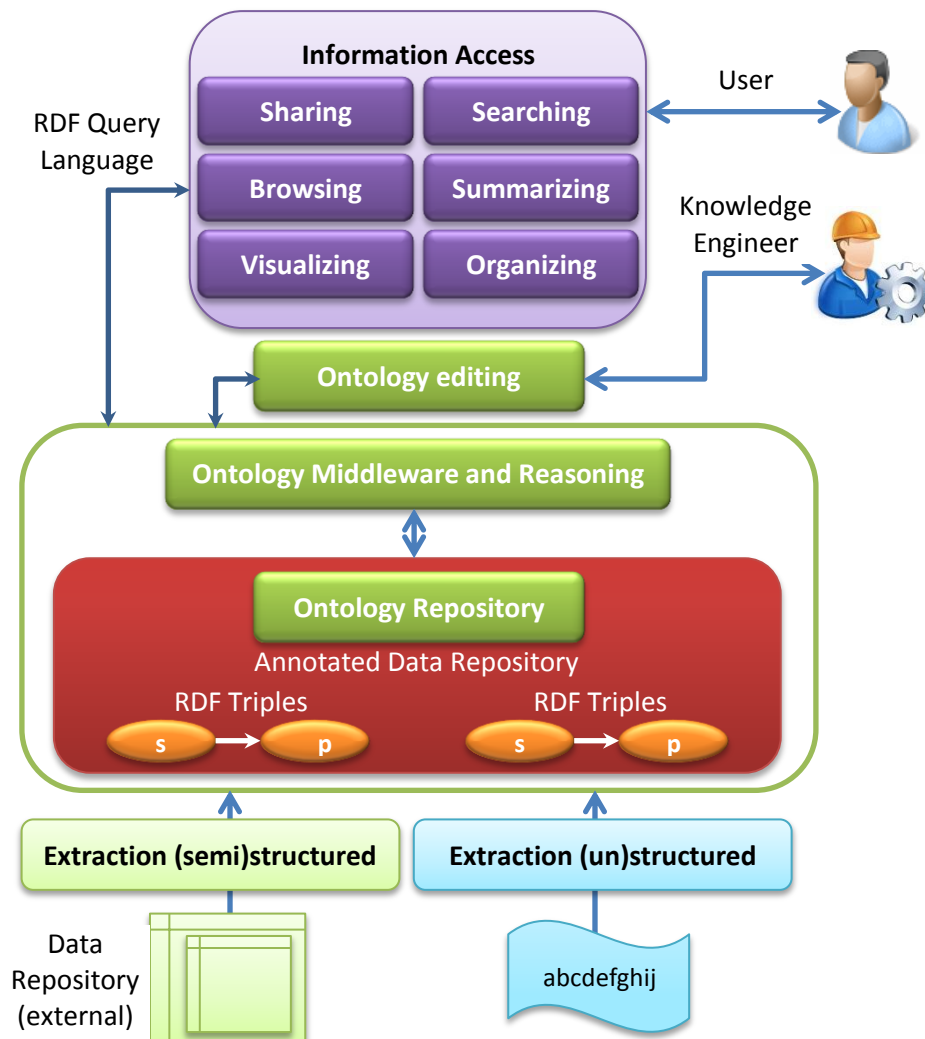


Figure 4. Semantic Web knowledge management architecture

Given the large amounts of unstructured and semi-structured information held on organizational intranets, automatic knowledge extraction from unstructured and semi-structured data in external data repositories is required and this is shown in the bottom layer of the diagram. Support for human knowledge acquisition is also needed and the knowledge engineer needs to be supported by ontology editing tools which support the creation, maintenance and population of ontologies. [16]

Once knowledge has been acquired from human sources or automatically extracted, it is then required to represent the knowledge in an ontology language (and of course to provide a query language to provide access to the knowledge so stored). This is the function of the ontology repository.

Ontology middleware is required with support for development, management, maintenance, and use of knowledge bases.

Finally, and perhaps most importantly, information access tools are required to allow end users to exploit the knowledge represented in the system. Such tools include facilities for finding, sharing, summarizing, visualizing, browsing and organizing knowledge.

3.1.4. Ontologies

Ontologies are a key enabling technology for the Semantic Web. They interweave human understanding of symbols with their machine processability.

The term ontology has its origin in philosophy and has been applied in many different ways. The word element onto- comes from the Greek ὄν, ὄντος «being; that which is», present participle of the verb εἶμι « be ». The core meaning within computer science is a model for describing the world that consists of a set of types, properties, and relationship types. Exactly what is provided around these varies, but they are the essentials of an ontology. There is also generally an expectation that there be a close resemblance between the real world and the features of the model in an ontology.

What many ontologies have in common in both computer science and in philosophy is the representation of entities, ideas, and events, along with their properties and relations, according to a system of categories. In both fields, one finds considerable work on problems of ontological relativity. [17]

Ontologies were developed in artificial intelligence to facilitate knowledge sharing and re-use. Since the early 1990s, ontologies have become a popular research topic. They have been studied by several artificial intelligence research communities, including knowledge engineering, natural-language processing and knowledge representation. More recently, the use of ontologies has also become widespread in fields such as intelligent information integration, cooperative information systems, information retrieval, electronic commerce, and knowledge management. The reason ontologies are becoming popular is largely due to what **they promise: a shared and common understanding of a domain that can be communicated between people and application systems**. As such, the use of ontologies and supporting tools offers an opportunity to significantly improve knowledge management capabilities in large organizations.

Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. As mentioned above, most ontologies describe individuals

(instances), classes (concepts), attributes, and relations. In this section each of these components is discussed in turn. Common components of ontologies include:

- Individuals: instances or objects (the basic or "ground level" objects)
- Classes: sets, collections, concepts, classes in programming, types of objects, or kinds of things
- Attributes: aspects, properties, features, characteristics, or parameters that objects (and classes) can have
- Relations: ways in which classes and individuals can be related to one another
- Function terms: complex structures formed from certain relations that can be used in place of an individual term in a statement
- Restrictions: formally stated descriptions of what must be true in order for some assertion to be accepted as input
- Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form
- Axioms: assertions (including rules) in a logical form that together comprise the overall theory that the ontology describes in its domain of application. This definition differs from that of "axioms" in generative grammar and formal logic. In those disciplines, axioms include only statements asserted as a priori knowledge. As used here, "axioms" also include the theory derived from axiomatic statements
- Events: the changing of attributes or relations

Ontologies are commonly encoded using ontology languages. The tools that have been developed in this area of research address three key aspects:

- Acquiring ontologies and linking them with large amounts of data. For reasons of scalability this process must be automated based on information extraction and natural language processing technology. For reasons of quality this process requires the human in the loop to build and manipulate ontologies using ontology editors.
- Storing and maintaining ontologies and their instances. A resource description framework (RDF) schema repository is developed that provides database technology and simple forms of reasoning over web information sources.
- Querying and browsing semantically enriched information sources.

3.2. Linked Open Data

In 2001 Tim Berners-Lee and his collaborators published a seminal article called “The Semantic Web” in which they presented their idea of “a new form of Web content that is meaningful to computers [and] will unleash a revolution of new possibilities”. In the last few years, the idea has gained traction and technologies have become available to build parts of this vision. Unfortunately, getting started is not so easy, because there are many concepts with slightly varying names and minute differences in their meaning and several technologies with cryptic names.

To begin with, what is the definition for *Linked Open Data* ? Linked Data by itself doesn’t have to be publicly available data, it can just as well be used in private, so we need one more definition: *Open Data*. It describes “a philosophy and practice requiring that certain data be freely available to everyone, without restrictions from copyright, patents or other mechanisms of control” (Wikipedia). This is similar in spirit to other movements like Open Source Software, and there is work being done to create licenses that clarify the usage terms of the data (e.g. Open Definition and the Open Data Commons).

Linked Open Data is used to describe the combination of data that is open and linked. This is the data we want, because it has clear license terms and is easily linkable with other data sets. These terms are put in relation to each other, in Figure 5. Relation of Data. [18]

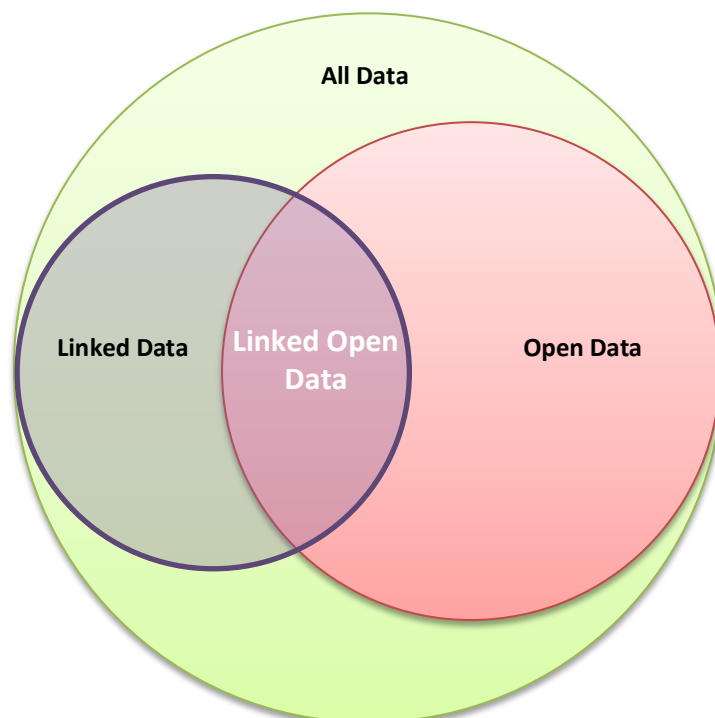


Figure 5. Relation of Data

Semantic Web and Linked Open Data

Governments have always had to make the data they produce transparent to their citizens, however, many do so using proprietary file formats like Excel, machine-unfriendly documents like PDFs, or “hide” the data by distributing it over many government sites and thus making it hard to find. This is all Open Data, because people can look at and use it.

The new trend now is to make data really open, not just legally but also as a matter of form. Sites have started to provide Open Data as a central, searchable catalog, often with the option of accessing the data through APIs, which makes it a lot easier to consume the data, as it doesn’t have to be transformed, combined and prepared for a program to use. With this central catalog in place, they have now been able to go a step further and start transforming this data into a huge Linked Open Data set, that is accessible to everyone. Figure 6. Linked Open Data dataset - September 2011, shows the size of the Linked Open Data web as of September 2011: each bubble is a website that you can access through Linked Open Data technologies in similar ways that you would normally access a database. [19]

Tim Berners-Lee suggested a 5-star deployment scheme for Linked Open Data and Ed Summers provided it, providing also useful examples as well as the costs and benefits of each category¹. [20]

- ★ make your stuff available on the Web (whatever format) under an open license
- ★★ make it available as structured data (e.g., Excel instead of image scan of a table)
- ★★★ use non-proprietary formats (e.g., CSV instead of Excel)
- ★★★★ use URIs to identify things, so that people can point at your stuff
- ★★★★★ link your data to other data to provide context

¹ <http://lab.linkeddata.deri.ie/2010/star-scheme-by-example/>

Semantic Web and Linked Open Data

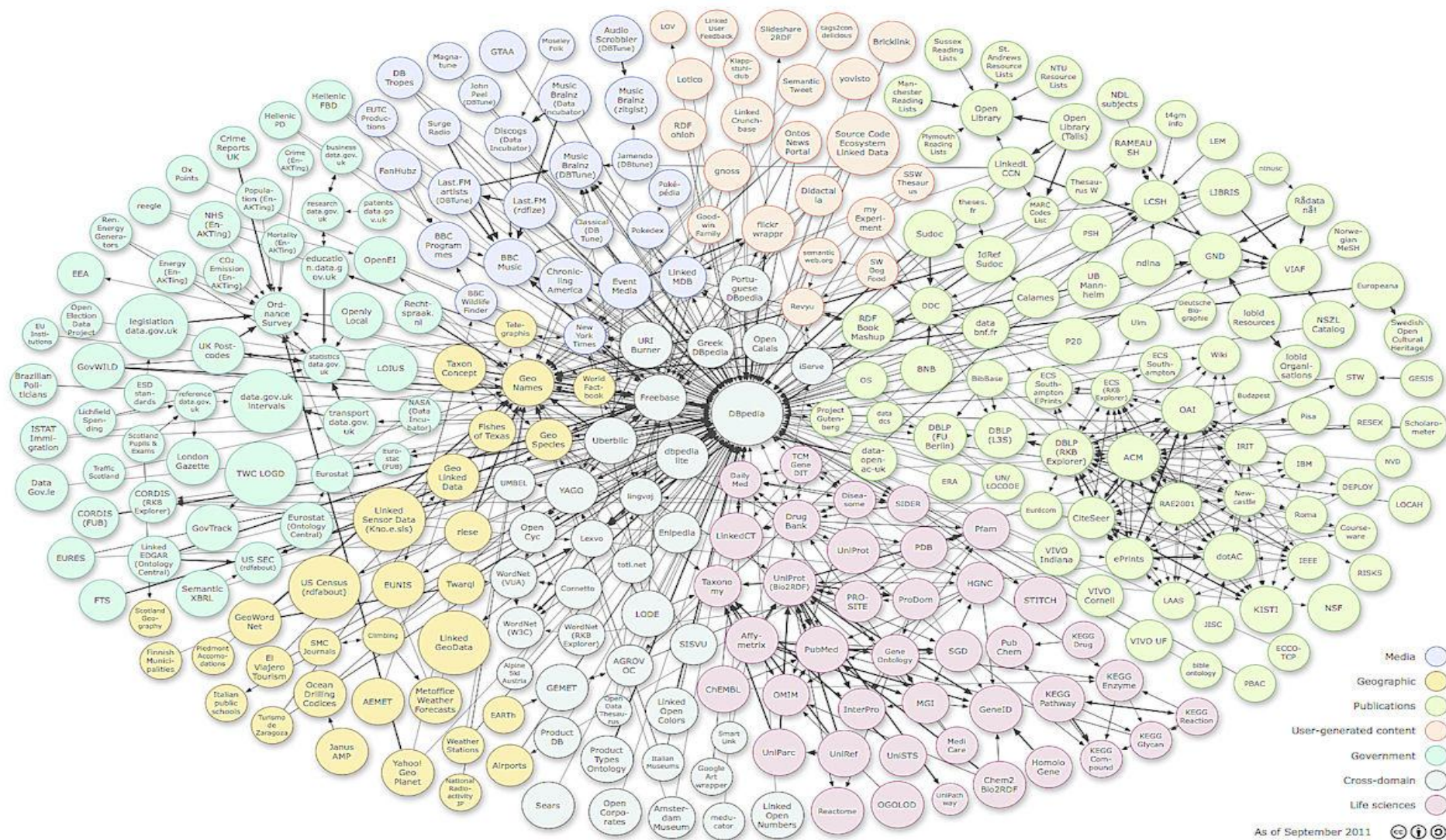


Figure 6. Linked Open Data dataset - September 2011 (source: [19])

Technologies and standards of the Semantic Web

Berners-Lee' suggested 5-star system describe the accessibility quality of data sets to emphasize that “the Semantic Web isn't just about putting data on the web”, but doing so in ways that allow machines to understand the meaning of the data.

What the system does not take into account, however, is the *quality* of the data itself. Bad content in, bad content out does still apply. This problem is especially acute for Linked Open Data at the moment, because everyone is just starting out with creating the ontologies and links and there is no way to do this overnight, so incompleteness will probably prevail for a while.

The benefits of Linked Open Data are numerous with a lot of value for society. Some of the most obvious benefits are:

- Because the data is freely accessible, it is verifiable by others, they can access and judge the source.
- Others can access the same data set and create alternative works.
- By linking up the data, we gain access to many more “databases” and discover connections we wouldn't otherwise have thought of.
- Because the knowledge about its relationships is already in the data, more people can work with it without requiring knowledge of tools to combine different data sets.
- The data becomes embeddable: not everyone has to keep a copy of a data set, they can link to it. This ensures that the data is always up-to-date.
- The data becomes better documented. If, for example, you use a national coordinate system, you can link to its definition and by that ease the difficulty for others to convert it to other systems.

The Linked Data paradigm has evolved from a practical research idea into a very promising candidate for addressing one of the biggest challenges in the area of intelligent information management: the exploitation of the Web as a platform for data and information integration in addition to document search. To translate this initial success into a world-scale disruptive reality, encompassing the Web 2.0 world and enterprise data alike, the following research challenges need to be addressed:

- Improve coherence and quality of data published on the Web,
- Close the performance gap between relational and RDF data management,
- Establish trust on the Linked Data Web and generally
- Lower the entrance barrier for data publishers and users. [6]

For this reason the European Commission launched in 2010 LOD2², a large-scale integrating project funded within the FP7 Information and Communication Technologies Work Programme. Commencing in September 2010, this 4-year project comprises leading Linked Open Data technology researchers, companies, and service providers (15 partners) from across 11 European countries (and one associated partner from Korea) and is coordinated by the AKSW research group at the University of Leipzig. With partners among those who initiated and strongly supported the Linked Open Data initiative, the LOD2 project aims at tackling these challenges by developing:

- enterprise-ready tools and methodologies for exposing and managing very large amounts of structured information on the Data Web,
- a testbed and bootstrap network of high-quality multi-domain, multi-lingual ontologies from sources such as Wikipedia and OpenStreetMap.
- algorithms based on machine learning for automatically interlinking and fusing data from the Web.
- standards and methods for reliably tracking provenance, ensuring privacy and data security as well as for assessing the quality of information.
- adaptive tools for searching, browsing, and authoring of Linked Data.

² <http://lod2.eu/Welcome.html>

4. Technologies and standards of the Semantic Web

4.1. Metadata

Metadata are structured data that describe the characteristics of an object. The prefix "meta" means a description in a higher level of a more basic one. A metadata record consists of a number of pre-certain elements representing specific attributes of an object/source and each component may take one or more values.

Each metadata schema typically has the following characteristics: a limited number of items, their names and their meanings. Usually, the semantic description refers to the contents, the space of existence, the physical properties, the type (e.g. text or picture, map or model) and form (e.g. printed copy, electronic file). Characteristics of metadata that support access to documents include author, date and place issued, and the topics that are covered. If the material of the information is a non-digital form (e.g. printed version) additional metadata are provided to help in the location of the material.

A metadata record consists of a certain number of pre-elements that represent specific properties of an object / source and each component may take one or more values as shown in the example that follows:

Element	Value
Title	Website
Creator	Web Development Team
Publisher	Department of Engineering and Computing
Identifier	www.ntua.gr
Type	Text / HTML
Relationship	University's Informative Website

Metadata (metacontent) syntax refers to the rules created to structure the fields or elements of metadata (metacontent). A single metadata scheme may be expressed in a number of different markups or programming languages, each of which requires a different syntax. For example, Dublin Core may be expressed in plain text, HTML, XML and RDF.

A common example of a metacontent is the bibliographic classification, the subject, the Dewey Decimal class number. There is always an implied statement in any "classification" of some object. To classify an object as, for example, Dewey class number 514 (Topology) (e.g. a book has this number on the spine) the implied statement is: "<book><subject heading><514>". This is a subject-predicate-

Technologies and standards of the Semantic Web

object triple, or more importantly, a class-attribute-value triple. The first two elements of the triple (class, attribute) are pieces of some structural metadata having a defined semantic. The third element is a value, preferably from some controlled vocabulary, some reference (master) data. The combination of the metadata and master data elements results in a statement which is a metacontent statement i.e. "metacontent = metadata + master data". All these elements can be thought of as "vocabulary". Both metadata and master data are vocabularies which can be assembled into metacontent statements. There are many sources of these vocabularies, both meta and master data: UML, EDIFACT, XSD, Dewey/UDC/LoC, SKOS, ISO-25964, Pantone, Linnaean Binomial Nomenclature etc. Using controlled vocabularies for the components of metacontent statements, whether for indexing or finding, is endorsed by ISO-25964: "If both the indexer and the searcher are guided to choose the same term for the same concept, then relevant documents will be retrieved." This is particularly relevant when considering that the behemoth of the internet, Google, is simply indexing then matching text strings, there is no intelligence or "inferencing" occurring.

The community that provides the information can also define a logical grouping of data or leave the process in each metadata schema. Although the syntax is not strictly part of the metadata schema, the data would be unusable unless the encoding schema understood the concepts of metadata schema. The encoding allows the metadata to be editable by a program. [2]

4.2. URLs, URIs, URNs, IRIs and Namespaces

When Berners-Lee invented the Web, along with writing the first web server and browser, he developed specifications for three things so that all the servers and browsers could work together:

- A way to represent document structure, so that a browser would know which parts of a document were paragraphs, which were headers, which were links, and so forth. This specification is the Hypertext Markup Language, or HTML.
- A way for client programs such as web browsers and servers to communicate with each other. The Hypertext Transfer Protocol, or HTTP, consists of a few short commands and three-digit codes that essentially let a client program such as a web browser request a file from the server respond by sending the file or presenting an unknown file message.
- A compact way for the client to specify which resource it wants—for example, the name of a file, the directory where it's stored, and the server that has that file system. This could be called a web address, or a resource locator. Berners-Lee called a server-directory-resource name combination that a client sends using a particular internet protocol (for example, <http://www.learningsparql.com/resources/index.html>) a Uniform Resource Locator, or URL. [4]

When you own a domain name like *worth2c.info* or *sensorit.info*, you control the directory structure and file names used to store resources there. This ability of a domain name owner to control the naming scheme (similarly to the way that Java package names build on domain names) led developers to use these names for resources that weren't necessarily web addresses. For example, the Friend of a Friend (FOAF) vocabulary uses <http://xmlns.com/foaf/0.1/Person> to represent the concept of a person, but if you send your browser to that “address,” it will just be redirected to the spec's home page.

This confused many people, because they assumed that anything that began with “http://” was the address of a web page that they could view with their browser. This confusion led two engineers from MIT and Xerox to write up a specification for Universal Resource Names, or URNs. A URN might take the form *urn:isbn:006251587X* to represent a particular book or *urn:schemasmicrosoft-com:office:office* to refer to Microsoft's schema for describing the structure of Microsoft Office files.

The term Universal Resource Identifier was developed to encompass both URLs and URNs. This means that a URL is also a URI. URNs didn't really catch on, though. So, because hardly anyone uses URNs, most URIs are URLs, and that's why people

sometimes use the terms interchangeably. It's still very common to refer to a web address as a URL, and it's fairly typical to refer to something like *http://xmlns.com/foaf/0.1/Person* as a URI instead, because it's just an identifier—even though it begins with “http://”.

As if this wasn't enough names for variations on URLs, the Internet Engineering Task Force released a spec for the concept of Internationalized Resource Identifiers. IRIs are URIs that allow a wider range of characters to be used in order to accommodate other writing systems. For example, an IRI can have Chinese or Cyrillic characters, and a URI can't. In general usage, “IRI” means the same thing as “URI.”

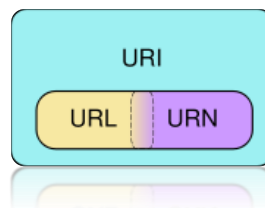


Figure 7. The relation between URI, URL and URN

URIs helped to solve the following problem. As the XML markup language became more popular, XML developers began to combine collections of elements from different domains to create specialized documents. This led to a difficult question: what if two sets of elements for two different domains use the same name for two different things? For example, if we want to say that Tim Berners-Lee's title at the W3C is “Director” and that the title of the book he wrote is “Weaving the Web,” I need to distinguish between these two senses of the word “title.” Computer science has used the term “namespace” for years to refer to a set of names used for a particular purpose, so the W3C released a spec describing how XML developers could say that certain terms come from specific namespaces. This way, they could distinguish between different senses of a word like “title.”

How do we name a namespace and refer to it? With a URI, of course. For example, the name for the Dublin Core standard set of basic metadata terms is the URI *http://purl.org/dc/elements/1.1/*. An XML document's main enclosing element often includes the attribute setting *xmlns:dc="http://purl.org/dc/elements/1.1/"* to indicate that the dc prefix will stand for the Dublin Core namespace URI in that document. Imagine that an XML processor found the following element in such a document:

```
<dc:title>Weaving the Web</dc:title>
```

It would know that it meant “title” in the Dublin Core sense—the title of a work.

Technologies and standards of the Semantic Web

The URIs that identify RDF resources are like the unique ID fields of relational database tables, except that they're universally unique, which lets you link data from different sources around the world instead of just linking data from different tables in the same database.

4.3. Resource Description Framework (RDF)

4.3.1. Introduction

The Resource Description Framework (RDF) was originally created in 1999 as a standard on top of XML for encoding metadata. Since then, and perhaps even after the updated RDF spec in 2004, the scope of RDF has really evolved into something greater. The most exiting uses of RDF aren't in encoding information about Web resources, but information about and relations between things in the real world: people, places, concepts, etc. [2]

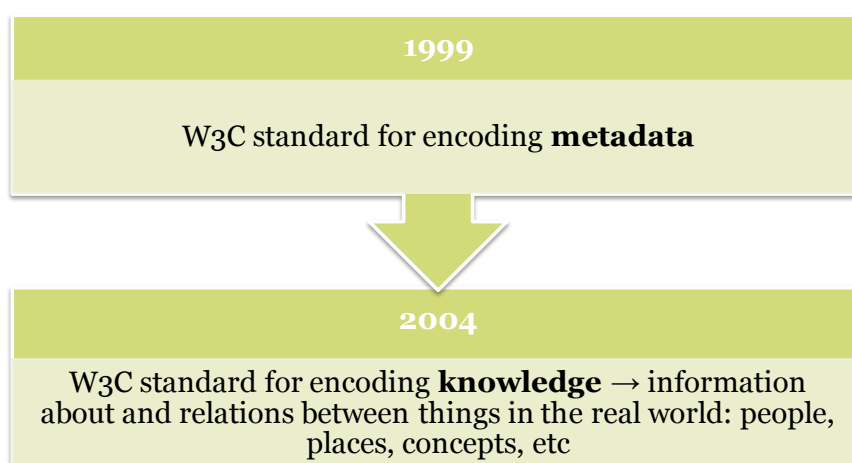


Figure 8. RDF evolution

Although often called a “language”, RDF is essentially a *data model*. Its basic building block is an object-attribute-value triple, called a *statement*. The example presented on page 30 with the representation of a professor teaching a specific course, is such a statement. Of course, an abstract data model needs a concrete syntax in order to be represented and transmitted, and RDF has been given a syntax in XML. As a result, it inherits the benefits associated with XML. However, it is important to understand that other syntactic representations of RDF, not based on XML, are also possible; XML-based syntax is not a necessary component of the RDF model.

RDF is domain-independent in that no assumptions about a particular domain of use are made. It is up to users to define their own terminology in a schema language called *RDF Schema (RDFS)*. The name RDF Schema is now widely regarded as an unfortunate choice. It suggests that RDF Schema has a similar relation to RDF as XML Schema has to XML, but in fact this is not the case. **XML Schema constrains the structure of XML documents, whereas RDF Schema defines the vocabulary used in RDF data models.** In RDFS we can define the vocabulary, specify which properties apply to which kinds of objects and what values they can take, and describe the relationships between objects. For example, we can write

Professor is a subclass of academic staff member.

This sentence means that all professors are also academic staff members. It is important to understand that there is an intended meaning associated with “is a subclass of”. It is not up to the application to interpret this term; its intended meaning must be respected by all RDF processing software. Through fixing the semantics of certain ingredients, RDF/RDFS enables us to model particular domains. [1]

RDF was designed primarily for situations where **information needs to be processed by applications, rather than just be presented in humans**. RDF provides a common framework for representing, such information in order to allow the exchange between different applications without misunderstandings and loss of accuracy. This feature makes the information available in other applications, different from those for which they were originally built. [2]

4.3.2. Basic Ideas of RDF

RDF is based on the idea of *identifying objects on the Web using specific identifiers, URIs, and the description of information sources using simple properties and values*. This enables RDF to represent simple statements about resources as a graph.

Such as HTML, the RDF/XML code is processed by machines and by use of URIs can link pieces of information of the entire Web. However, unlike conventional hypertext, RDF URIs can refer to any object that can be identified, including items that cannot be returned directly through the internet. As a consequence, in addition to the description of objects, such as Web pages, RDF can also describe cars, businesses, people, news, etc. Still, RDF properties themselves have URIs to identify the precise relationships between related objects. [2]

The fundamental concepts of RDF are resources, properties, and statements.

Resources

We can think of a resource as an object, a “thing” we want to talk about. Resources may be authors, books, publishers, places, people, hotels, rooms, search queries, and so on. Every resource has a URI, a Uniform Resource Identifier. A URI can be a URL (Uniform Resource Locator, or Web address) or some other kind of unique identifier; note that an identifier does not necessarily enable *access* to a resource. URI schemes have been defined not only for Web locations but also for such diverse objects as telephone numbers, ISBN numbers, and geographic locations. There has been a long discussion about the nature of URIs, even touching philosophical questions (for

Technologies and standards of the Semantic Web

example, what is an appropriate unique identifier for a person?). In general, we assume that a URI is the identifier of a Web resource.

Properties

Properties are a special kind of resources; they describe relations between resources, for example “written by”, “age”, “title”, and so on. Properties in RDF are also identified by URIs (and in practice by URLs). This idea of using URIs to identify “things” and the relations between them is quite important. This choice gives us in one stroke a global, worldwide, unique naming scheme. The use of such a scheme greatly reduces the homonym problem that has plagued distributed data representation until now. *In RDF predicates (properties) can have only binary values.*

Statements

Statements assert the properties of resources. A statement is an object-attribute-value triple, consisting of a resource, a property, and a value. Values can either be resources or *literals*. Literals are atomic values (strings). [1]

4.3.3. Triple view of RDF statements

In the RDF model, information is represented by a set of statements that consist of the three parts described in section 4.3.2. Basic Ideas. Because of their structure, the RDF statements are called **triples**. The three components of a statement have a similar role to that: the subject indicates the person or thing to which the statement and the predicate describe a relationship between subject and object (1st view). In Figure 9. Graphical representation of an RDF Triple, we see a graph-based view of a statement (2nd view). It is a directed graph with labeled nodes and arcs; the arcs are directed from the resource (the *subject* of the statement) to the value (the *object* of the statement). This kind of graph is known in the Artificial Intelligence community as a *semantic net*. [1]

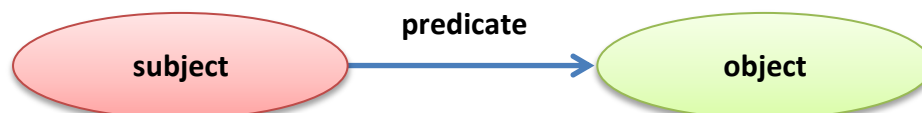


Figure 9. Graphical representation of an RDF Triple

An example of such a triple could be the following



Figure 10. Example of an RDF Triple

As it was stated earlier, the value of a statement can be a resource. Therefore, it may be linked to other resources, as in the following example.

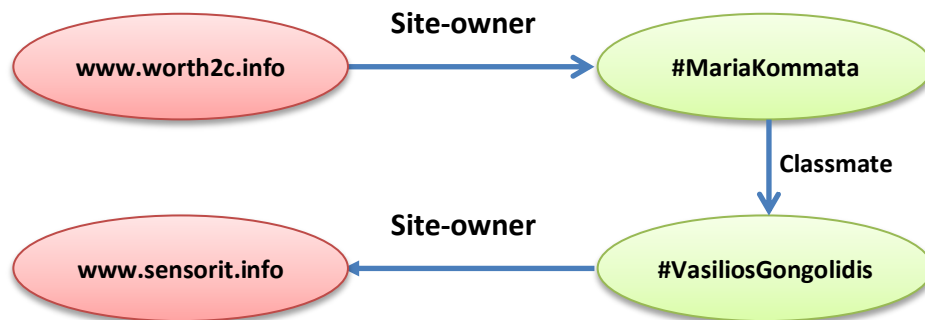


Figure 11. Example of combined graphs

Graphs are a powerful tool for human understanding. But the Semantic Web vision requires machine-accessible and machine-processable representations. Therefore, there is another representation possibility (the 3rd) based on XML. According to this possibility, an RDF document is represented by an XML element with the tag `rdf:RDF`. The content of this element is a number of *descriptions*, which use `rdf:Description` tags. Every description makes a statement about a resource, which is identified in one of three different ways:

- An `about` attribute, referencing an existing resource
- An `ID` attribute, creating a new resource
- Without a name, creating an anonymous resource

According to the above, the representation of the statement that is depicted in Figure 10. Example of an RDF Triple, would be:

```
<?xml version="1.0" encoding="UTF-16"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:mydomain="http://www.mydomain.org/my-rdf-ns">

  <rdf:Description rdf:about="http://www.worth2c.info">
    <mydomain:site-owner rdf:resource="#MariaKommata"/>
  </rdf:Description>
</rdf:RDF>
```

The first line specifies that we are using XML.

The `rdf:Description` element makes a statement about the resource `http://www.worth2c.info`. Within the description the property is used as a tag, and the content is the value of the property.

The descriptions are given in a certain order; in other words, the XML syntax imposes a *serialization*. The order of descriptions (or resources) is *not* significant according to the abstract model of RDF. This again shows that the graph model is the real data model of RDF and that XML is just a possible serial representation of the graph.

However, there are two barriers for the communication of applications using RDF. First, there should be various *online system resource identifiers* to avoid confusion about which subject, object or predicate use. There should be identifiers in order to avoid confusion between the communicating parties. Secondly, a *common language* should be set, easily processable by computers to represent the declarations in order to set the communication between them easier.

The solution of these two issues is given by the existing Internet infrastructure. The issue of identifiers for network resources is easily solved by using URIs (see section 4.2. URLs, URIs, URNs, IRIs and Namespaces). In fact, RDF uses URI references (URI references or URIref). A URIref is a URI with an optional identifier part to the end. The URIrefs of RDF can contain Unicode characters, allowing multiple languages to be represented in URIrefs. RDF defines a resource as anything that is identifiable through an URIref. Therefore, URIrefs allow RDF to describe everything, even to state the relationship between objects.

As for the subject of common mechanical editable language, XML is the best solution. XML enables anyone to create his own configuration and data model. However, its usefulness is limited to local data transactions among companies, because the interpretation of XML documents must be coded within the application. The RDF provides exactly what is missing from the XML, and composes with it a neoformat the RDF/XML.

4.3.3.1. *RDF: XML-Based syntax*

The namespace mechanism of XML is used in RDF, but in an expanded way. In XML namespaces are only used for disambiguation purposes. In RDF external namespaces are expected to be RDF documents defining resources, which are then used in the importing RDF document. This mechanism allows the reuse of resources by other people who may decide to insert additional features into these resources. The result is the emergence of large, distributed collections of knowledge.

The `rdf:about` attribute of the element `rdf:Description` is equivalent in meaning to that of an ID attribute, but it is often used to suggest that the object about which a statement is made has already been “defined” elsewhere. Formally speaking, a set of RDF statements together simply forms a large graph, relating things to other things through properties, and there is no such thing as “defining” an object in one place and referring to it elsewhere. The property elements of a description must be read conjunctively.

The attribute `rdf:datatype` is used to indicate the data type of the value of a resource property. It is required to indicate the type of the value of this property each time it is used. This is to ensure that an RDF processor can assign the correct type of the property value even if it has not seen the corresponding RDF Schema definition before (a scenario that is quite likely to occur in the unrestricted World Wide Web). [1]

Container elements are used to collect a number of resources or attributes about which we want to make statements *as a whole*. For example, we may wish to talk about the courses given by a particular lecturer. Three types of containers are available in RDF:

- `rdf:Bag`, an unordered container, which may contain multiple occurrences (not true for a set). Typical examples are members of the faculty board and documents in a folder —examples where an order is not imposed.
- `rdf:Seq`, an ordered container, which may contain multiple occurrences. Typical examples are the modules of a course, items on an agenda, an alphabetized list of staff members—examples where an order is imposed.
- `rdf:Alt`, a set of alternatives. Typical examples are the document home and mirrors, and translations of a document in various languages.

Sometimes we wish to make statements about other statements. To do so we must be able to refer to a statement using an identifier. RDF allows such reference through a *reification* mechanism, which turns a statement into a resource. For example, the description

```
<rdf:Description rdf:about="949352">
  <uni:name>Maria Kommata</uni:name>
</rdf:Description>
```

reifies as

```
<rdf:Statement rdf:about="StatementAbout949352">
  <rdf:subject rdf:resource="949352"/>
  <rdf:predicate rdf:resource="&uni:name"/>
  <rdf:object> Maria Kommata </rdf:object>
</rdf:Statement>
```

The `rdf:subject`, `rdf:predicate`, and `rdf:object` allow us to access the parts of a statement. The ID of the statement can be used to refer to it, as can be done for any description. We can either write an `rdf:Description` if we don't want to talk about it further, or an `rdf:Statement` if we wish to refer to it.

4.3.3.2. *RDF versus XML*

In some ways, RDF can be compared to XML. XML also is designed to be simple and applicable to any type of data. XML is also more than a file format. It is a foundation for dealing with hierarchical, self-contained documents, whether they be stored on disk in the usual brackets-and-slashes format, or held in memory and accessed through a DOM API. [2]

What sets RDF apart from XML is that RDF is designed to represent knowledge in a distributed world. The fact that RDF is designed for knowledge, and not data, means that RDF is particularly concerned with meaning. Everything at all mentioned in RDF means something. It may be a reference to something in the world, like a person or movie, or it may be an abstract concept, like the state of owing something. And by putting three such entities together, the RDF standard says how to arrive at a fact. The meaning of the triple "(John, Pluto, the state of owing)" might be that John owns Pluto. By putting a lot of facts together, one arrives at some form of knowledge. Standards built on top of RDF add to RDF semantics for drawing logical inferences from data. Standards built on RDF describe logical inferences between facts and how to search for facts in a large database of RDF knowledge.

For comparison, XML itself is not very much concerned with meaning. XML nodes don't need to be associated with particular concepts, and the XML standard doesn't indicate how to derive a fact from a document. For instance, if you were presented with a few XML documents whose root nodes were in a foreign language you don't understand, you couldn't do anything useful with the documents but display them. RDF documents with nodes you can't understand could still actually be usefully processed because RDF specifies some basic level of meaning. Now, this isn't to say that you couldn't develop your own standard on top of XML that says how to derive the set of facts in an XML document, but you'll find you've probably just reinvented something like RDF.

The second key aspect of RDF is that it works well for distributed information. That is, RDF applications can put together RDF files posted by different people around the Internet and easily learn from them new things that no single document asserted. It does this in two ways, first by linking documents together by the common vocabularies they use, and second by allowing any document to use any vocabulary. This allows enormous flexibility in expressing facts about a wide range of things, drawing on information from a wide range of sources.

So why use RDF rather than XML? The answer is twofold:

- Firstly, the information in RDF maps directly and unambiguously to a model, a model which is decentralized, and for which there are many generic parsers already available. This means that when you have an RDF application, you know which bits of data are the semantics of the application, and which bits is just syntactic fluff. And not only do you know that, everyone knows that, often implicitly without even reading a specification because RDF is so well known.
- Secondly, we hope that RDF data will become a part of the Semantic Web, so the benefits of drafting your data in RDF now draws parallels with drafting your information in HTML in the early days of the Web.

4.3.3.3. *RDF Serialization Formats*

There are many different notations for writing (serializing) RDF triples. While the data model that RDF uses is very simple, the serialized representation tends to get complicated when an RDF graph is saved to a file or sent over a network because of the various methods used to compact the data while still leaving it readable. These compaction mechanisms generally take the form of shortcuts that identify multiple references to a graph node using a shared but complex structure.

4.3.3.3.1. *N-Triples*

N-Triple notation is a very simple but verbose serialization. Because of their simplicity, N-Triples were used by the W3C Core Working Group to unambiguously express various RDF test-case data models while developing the updated RDF specification. This simplicity also makes the N-Triple format useful when hand-crafting datasets for application testing and debugging. Each line of output in N-Triple format represents a single statement containing a subject, predicate, and object followed by a dot. Except for blank³ nodes and literals, subjects, predicates, and objects are expressed as absolute URIs enclosed in angle brackets.

³ In RDF, a blank node (also called *bnode*) is a node in an RDF graph representing a resource for which a URI or literal is not given. The resource represented by a blank node is also called an anonymous

Object literals are double-quoted strings that use the backslash to escape double-quotes, tabs, newlines, and the backslash character itself. String literals in N-Triple notation can optionally specify their language when followed by `@lang`, where `lang` is an ISO 639 language code. Literals can also provide information about their datatype when followed by `^^type`, where `type` is commonly an XSD (XML Schema Definition) datatype. [12]

The extension `.nt` is typically used when N-Triples are stored in a file, and when they are transmitted over HTTP the mime type `text/plain` is used. The official N-Triple format is documented at <http://www.w3.org/TR/rdf-testcases/#ntriples>. [21]

The following is an example of an N-Triple:

```
<http://www.w3.org/2001/sw/RDFCore/ntriples/>
<http://purl.org/dc/elements/1.1/publisher>
<http://www.w3.org/>
```

4.3.3.3.2. Notation 3 (N3) syntax

While N-Triples are conceptually very simple, a lot of repetition in the output. The redundant information takes additional time to transmit and parse. While it's not a problem when working with small amounts of data, the additional information becomes a liability when working with large amounts of data. By adding a few additional structures, N3 condenses much of the repetition in the N-Triple format. [22]

In an RDF graph, every connection between nodes represents a triple. Since each node may participate in a large number of relationships, we could significantly reduce the number of characters used in N-Triples if we used a short symbol to represent repeated nodes. We could go further, recognizing that many of the URIs used in a specific model frequently come from related URIs. In much the same way that XML provides a namespace mechanism for generating short Qualified Name (qnames) for nodes, N3 allows us to define a URI prefix and identify entity URIs relative to a set of prefixes declared at the beginning of the document. The statement:

```
PREFIX ab: <http://learningsparql.com/ns/addressbook#>.
```

resource. By RDF standard a blank node can only be used as subject or object in an RDF triple, although in some syntaxes like Notation 3 it is acceptable to use a blank node as a predicate.

allows us to shorten the absolute URI for DuCharme⁴ from `<http://learningsparql.com/ns/addressbook#/DuCharme>` to `ab:DuCharme`.

Since each node in an RDF graph is a potential subject about which we may have many things to say, it is not uncommon to see the same subject repeat many (many) times in N-Triple output. N3 reduces this repetition by allowing the combination of multiple statements about the same subject by using a semicolon (;) after the first statement; After that only the predicate and object for other statements using the same subject needs to be stated.

The following statement says that DuCharme knows Toby and that DuCharme's email address is `ducharme@oreilly.com` (note how `ab:DuCharme`, the subject, is only stated once):

```
ab:DuCharme foaf:knows <http://kiwitobes.com/toby.rdf#ts>;
foaf:mbox "ducharme@oreilly.com".
```

N3 also provides a shortcut that allows you to express a group of statements that share a common anonymous subject (blank node) without having to specify an internal name for the blank node. Mailing addresses are frequently modeled with a blank node to hold all the components of the address together. W3C has defined a vocabulary for representing the data elements of the vCard interchange format that includes predicates for modeling street addresses. For instance, to specify the address of O'Reilly, you could write:

```
[ <http://www.w3.org/2006/vcard/ns#street-address> "1005
Gravenstein Hwy North" ;
<http://www.w3.org/2006/vcard/ns#locality> "Sebastopol,
California"
].
```

Because it is important to explicitly state that an entity is of a certain type, N3 allows you to use the letter `a` as a predicate to represent the RDF "type" relationship represented by the URI `<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>`.

Another predicate for which N3 provides a shortcut is `<http://www.w3.org/2002/07/owl#sameAs>`. OWL (Web Ontology Language) is a vocabulary for defining precise relationships between model elements.

⁴ Author of the book "Learning SPARQL", O'Reilly, 2011

Technologies and standards of the Semantic Web

You will frequently see the `owl:sameAs` predicate to express that two URIs refer to the same entity. The `sameAs` predicate is used so frequently that the authors of N3 designated the symbol `=` as shorthand for it. Because N-Triples are a subset of N3, any library capable of reading N3 will also read N-Triples.

4.3.3.3. RDF/XML

The original W3C Recommendation on RDF covered both a description of RDF as a data model and XML as an expression of RDF models. Because of this, people sometimes refer to RDF/XML as RDF, but it is important to recognize that it is just one possible representation of an RDF graph. RDF/XML is sometimes criticized for being difficult to read due to all the abbreviated structures it provides; still, it is one of the most frequently used formats, so it's useful to have some familiarity with its layout. [12]

Conceptually, RDF/XML is built up from a series of smaller descriptions, each of which traces a path through an RDF graph. These paths are described in terms of the nodes (subjects) and the links (predicates) that connect them to other nodes (objects). This sequence of “node, link, node” is repeatable. Since each node encountered in these descriptions has a strong identifier, it is possible to weave the smaller descriptions together to learn the larger RDF graph structure (see example in Figure 12. An example of a FOAF graph).

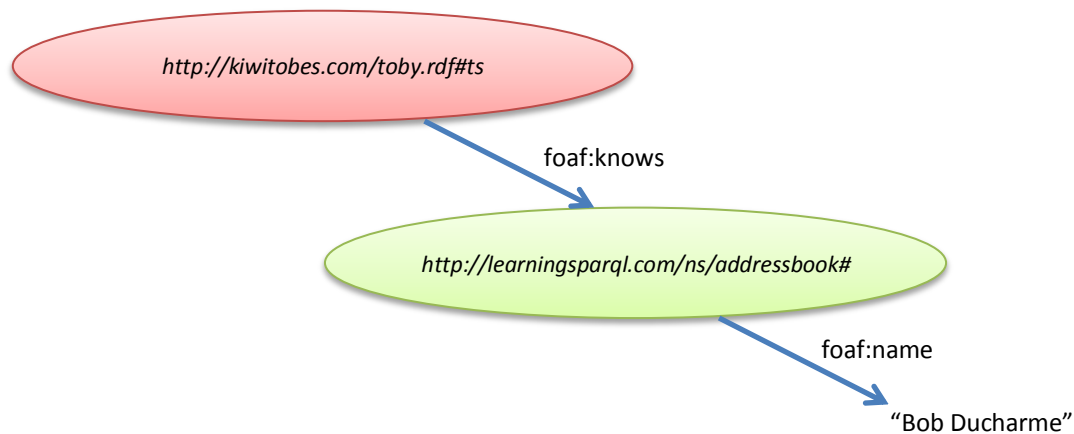


Figure 12. An example of a FOAF graph

If there is more than one path described in an RDF/XML document, all the descriptions must be children of a single RDF element; if there is only one path described, the `rdf:RDF` element may be omitted. As with other XML documents, the top-level element is frequently used to define other XML namespaces used throughout the document:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" />
```


Paths are always described starting with a graph node, using an `rdf:Description` element. The URI reference for the node can be specified in the description element with an `rdf:about` attribute. For blank nodes, a local identifier (valid only within the context of the current document) can be specified using an `rdf:NodeID` attribute. Predicate links are specified as child elements of the `rdf:Description` node, which will have their own children representing graph nodes.

4.3.3.3.4. Turtle (Terse RDF Triple Language)

Turtle is a textual syntax for RDF and it allows RDF graphs to be completely written in a compact and natural text form, with abbreviations for common usage patterns and datatypes. Turtle was defined by Dave Beckett and is intended to be compatible with, and a subset of, Notation 3 and is generally usable in systems that support N3.



It is actually an extension of N-Triples carefully taking the most useful and appropriate things added from N3 while keeping it in the RDF model. [2] [23]

The restrictions applied in RDF/XML that prevent it encoding all RDF graphs do not apply to Turtle. All RDF written in Turtle should be usable inside the query language part of the SPARQL (see section 4.7. SPARQL) which uses a Turtle/N3 style syntax for the Triple patterns and for RDF triples in the CONSTRUCT clause. This allows using RDF written in Turtle to allow forming "queries by example", using the data to make an initial query which can then be edited to use variables where bindings are wanted.

A statement in Turtle consists of a sequence of directives, triple-generating statements or blank lines. Comments may be given after a `"#"` and continue to the end of the line. The syntax is similar to N-Triples, where simple (s, p, o) triples are used, terminated by `"."`.

There are **three types of RDF Terms**, which are studied next: RDF URI References (URIs), literals and blank nodes.

In Turtle, **URIs** are written enclosed in `"<"` and `">"` and may be absolute RDF URI References or relative to the current base URI. URIs can be abbreviated as prefixed names, by using Turtle's `@prefix` directive that allows declaring a short prefix name for a long prefix of repeated URIs. This is useful for many RDF vocabularies that are all defined in nearby namespace URIs, possibly using XML's namespace mechanism that works in a similar fashion. Once a prefix such as `PREFIX ab: http://learningsparql.com/ns/addressbook#` is defined, any mention of a URI later in the document may use a qualified name that starts with `ab:` to stand for the longer

URI. So for example, the qualified name *ab:DuCharme* is a shorthand for the URI *http://learningsparql.com/ns/addressbook#DuCharme*.

Literals can be either simple, when they do not contain linebreaks, or long, when they may contain linebreaks. Literals may be given either a language suffix or a datatype URI but not both. Languages are indicated by appending the simple literal with @ and the language tag. Datatype URIs similarly append ^^ followed by any legal URI form (full or qualified) to give the datatype URI. Common datatypes are:

- Decimal integers may be written directly and correspond to the XML Schema Datatype `xsd:integer` in both syntax and datatype URI.
- Decimal floating point double/fixed precision numbers may be written directly and correspond to the XML Schema Datatype `xsd:double` in both syntax and datatype URI.
- Decimal floating point arbitrary precision numbers may be written directly and correspond to the XML Schema Datatype `xsd:decimal` in both syntax and datatype URI.
- Boolean may be written directly as `true` or `false` and correspond to the the XML Schema Datatype `xsd:boolean` in both syntax and datatype URI.

Blank nodes in Turtle, are written as `_:nodeID` to provide a blank node either from the given nodeID and may also be made with `[]` to provide the subject of RDF triples for each pair from the predicateObjectList or the root of the collection.

An example of an RDF serialization in Turtle is as follows.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  ex:editor [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

4.3.3.3.5. JSON (JavaScript Object Notation)

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, [Standard ECMA-262 3rd Edition - December 1999](#). JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language [24]. JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures. In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by: (colon) and the name/value pairs are separated by , (comma).

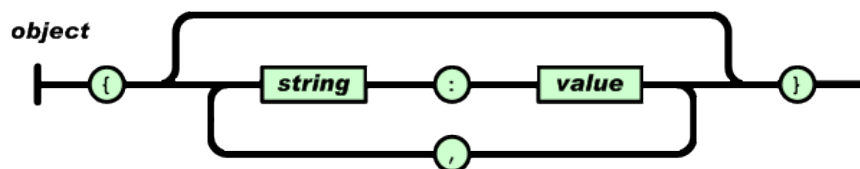


Figure 13. JSON object representation

An *array* is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).

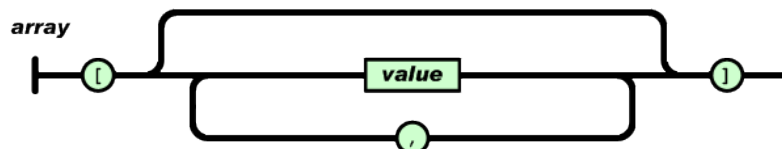


Figure 14. JSON array representation

A *value* can be a *string* in double quotes, or a *number*, or true or false or null, or an *object* or an *array*. These structures can be nested.

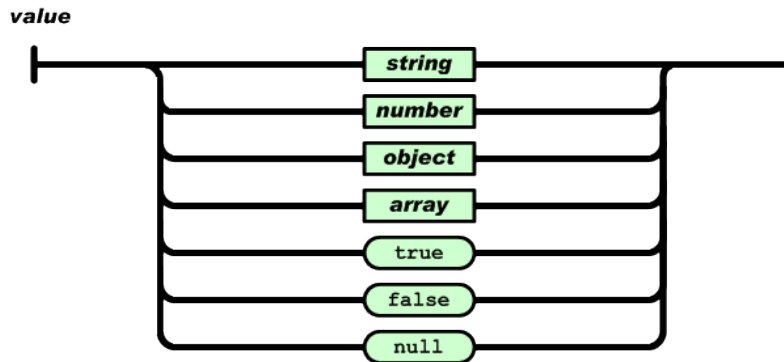


Figure 15. JSON value representation

A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.

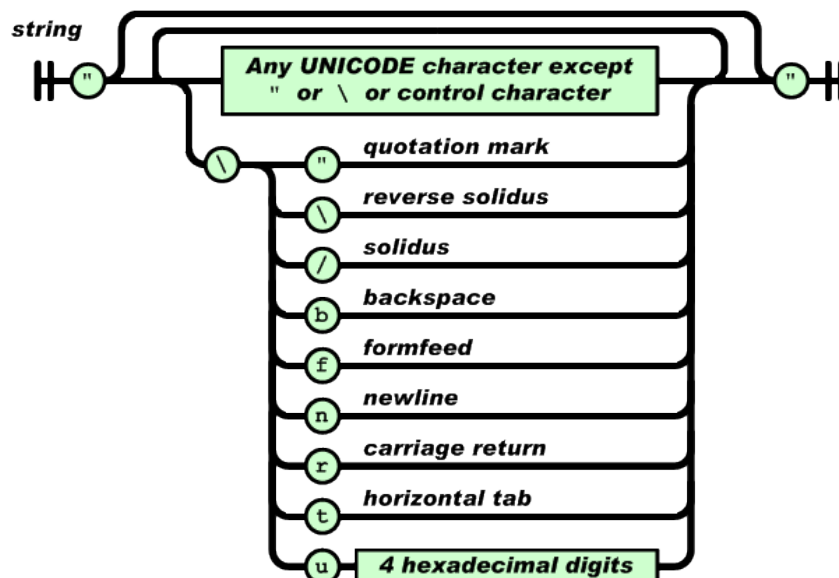


Figure 16. JSON string representation

A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used.

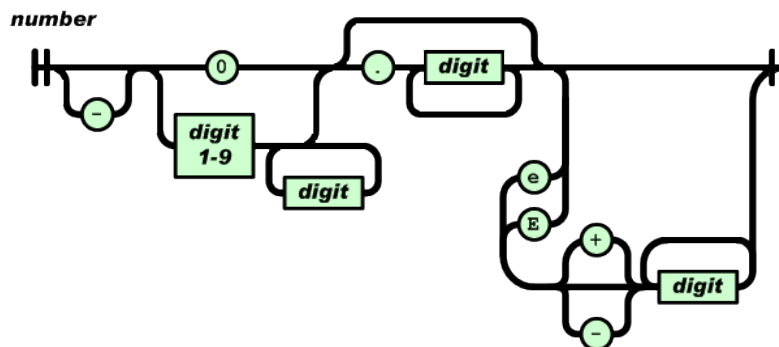


Figure 17. JSON number representation

Here is an example of an RDF serialization with JSON.

```

{
  "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized
Markup Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to
create markup languages such as DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}

```

Some of the goals that we are trying to achieve with a JSON serialization of RDF are the following:

- Support for scripting languages: Provide better support for processing RDF in scripting languages
- Creating convergence: Build some convergence around the dizzying array of existing RDF in JSON proposals, to create consistency in how data is published
- Gaining traction: Make RDF more acceptable for web developers, with the hope of increasing engagement with RDF and Linked Data

There are several drawbacks to the RDF/XML syntax that was described earlier.

Firstly, there is a mismatch in the data models: serialization involves turning a graph into a tree. There are many different ways to achieve that so, without applying some

Technologies and standards of the Semantic Web

external constraints, the output can be highly variable. The problem is that those constraints can be highly specific, so are difficult to generalize. This results in a high degree of syntax variability of RDF/XML in the wild, and that undermines the ability to use RDF/XML with standard XML tools like XPath, XSLT, etc. They operate only on the surface XML syntax not the “real” data model.

Secondly, because of the mismatch in (loosely speaking) the native data types. XML is largely about elements and attributes whereas RDF has resources, properties, literals, blank nodes, lists, sequences, etc. And of course there are those ever present URIs. This leads to additional syntax shortcuts and hijacking of features like XML namespaces to simplify the output, whilst simultaneously causing even more variability in the possible serializations.

Thirdly, when it comes to parsing, RDF/XML just isn’t a very efficient serialization. It’s typically more verbose and can involve much more of a memory overhead when parsing than some of the other syntaxes.

Because of these issues, we end up with a syntax which, while flexible, requires some profiling to be really useful within an XML tool chain. Or you just ignore the fact that it’s XML at all and throw it straight into a triple store. If you do that then an XML serialization of RDF is just a convenient way to *generate* RDF data from an XML tool chain.

Unfortunately when we look at serializing RDF as JSON we discover that we have nearly all of the same issues. JSON is a tree; so we have the same variety of potential options for serializing any given graph. The data types are also still different: key-value pairs, hashes, lists, strings, dates, etc. versus resource, properties, literals, etc. While there is potential to use more native datatypes, the practical issues of repeatable properties, blank nodes, etc mean that a 1:1 mapping isn’t feasible. Lack of support for anything like XML Namespaces means that hiding URIs is also impossible without additional syntax conventions. So, ultimately, both XML and JSON are poor fits for handling RDF. [25]

4.4. RDF Schema

RDF is a universal language that lets users describe resources using their own vocabularies. RDF does not make assumptions about any particular application domain, nor does it define the semantics of any domain. Is it up to the user to do so in RDF Schema (RDFS).

4.4.1. Classes

How do we describe a particular domain? Let us consider the domain of courses and lecturers at NTUA University. First we have to specify the “things” we want to talk about. Here we make a first, fundamental distinction. On one hand, we want to talk about particular lecturers, such as Vasilios Loumos (see also Section 2.3. Problems of today’s web) and particular courses, such as Multimedia Technology. We want to talk about courses, first-year courses, lecturers, professors, and so on. What is the difference with the approaches we have seen so far? In the first case we talk about *individual objects* (resources), in the second we talk about *classes* that define types of objects.

A class can be thought of as a set of elements. Individual objects that belong to a class are referred to as *instances* of that class. An important use of classes is to impose restrictions on what can be stated in an RDF document using the schema. [1]

4.4.2. Class Hierarchies and Inheritance

Once we have classes we would also like to establish relationships between them. For example, suppose that we have classes for

staff members	assistant professors
academic staff members	administrative staff members
professors	technical support staff members
associate professors	

These classes are not unrelated to each other. For example, every professor is an academic staff member. We say that “professor” is a subclass of “academic staff member”, or equivalently, that “academic staff member” is a superclass of “professor”. The subclass relationship defines a hierarchy of classes, as shown in Figure 18. RDF Schema - A hierarchy of classes.

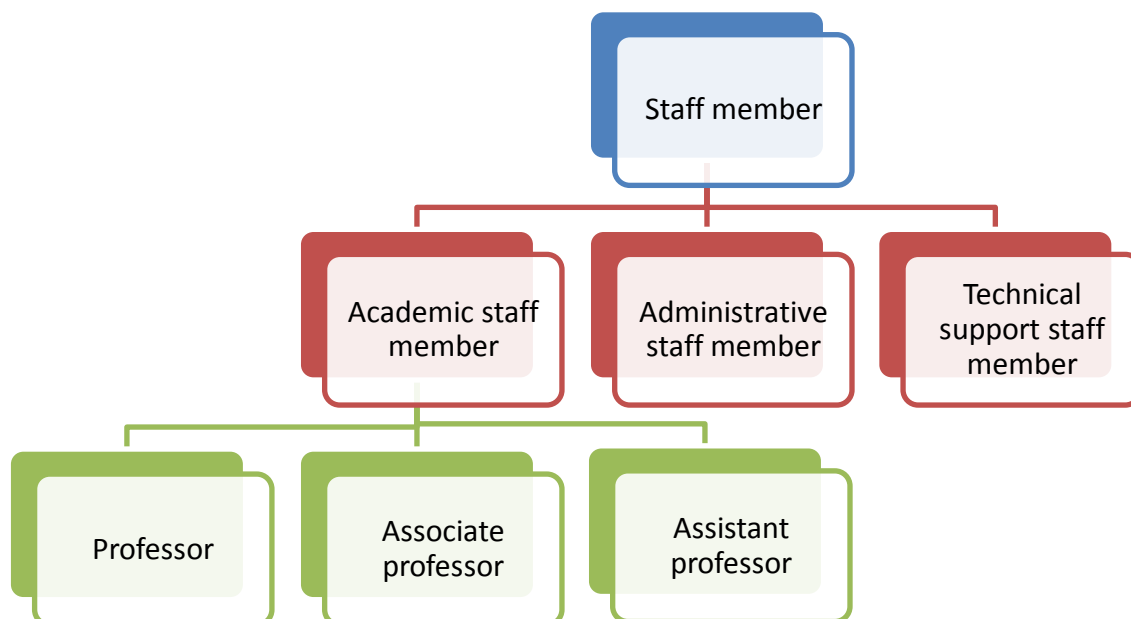


Figure 18. RDF Schema - A hierarchy of classes

There is no requirement in RDF Schema that the classes together form a strict hierarchy. In other words, a subclass graph as in Figure 18. RDF Schema - A hierarchy of classes need not be a tree. A class may have multiple superclasses. If a class *A* is a subclass of both *B1* and *B2*, this simply means that every instance of *A* is both an instance of *B1* and an instance of *B2*.

A hierarchical organization of classes has a very important practical significance, which we outline now. Consider the range restriction

Courses must be taught by academic staff members only.

Suppose Agis Papantoniou is defined as a professor. Then, according to the preceding restriction, he is not allowed to teach courses. The reason is that there is no statement specifying that Agis Papantoniou is also an academic staff member. It would be counterintuitive to overcome this difficulty by adding that statement to our description. Instead we would like Agis Papantoniou to *inherit* the ability to teach from the class of academic staff members. Exactly this is done in RDF Schema.

By doing so, RDF Schema *fixes the semantics* of “is a subclass of”. Now it is not up to an application to interpret “is a subclass of”; instead its intended meaning must be used by all RDF processing software. By making such semantic definitions RDFS is a (still limited) language for defining the semantics of particular domains. Stated another way, RDF Schema is a primitive *ontology language*. [1]

Technologies and standards of the Semantic Web

Classes, inheritance, and properties are, of course, known in other fields of computing, for example, in object-oriented programming. But while there are many similarities, there are differences, too. In object-oriented programming, an object class defines the properties that apply to it. To add new properties to a class means to modify the class.

However, in RDFS, properties are defined globally, that is, they are not encapsulated as attributes in class definitions. It is possible to define new properties that apply to an existing class without changing that class.

4.5. Web Ontology Language (OWL)

4.5.1. Requirements for Ontology Languages

A chronological representation of the evolution from RDF to OWL and of the technologies that emerged after RDF, is represented in Figure 19. Evolution from RDF to OWL. [26]

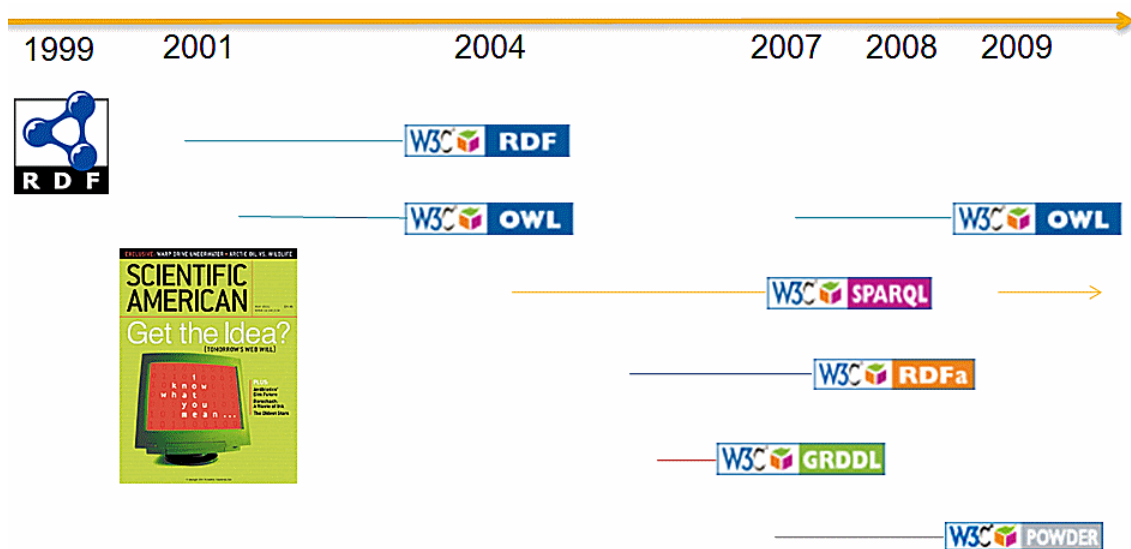


Figure 19. Evolution from RDF to OWL (source [26])

Ontology languages allow users to write explicit, formal conceptualizations of domain models. The main requirements are:

- a well-defined syntax,
- efficient reasoning support,
- a formal semantics,
- sufficient expressive power and
- convenience of expression.

The importance of a well-defined syntax is clear and known from the area of programming languages; it is a necessary condition for machine processing of information. All the languages that have been presented so far have a well-defined syntax.

Of course, it is questionable whether the XML-based RDF syntax is very user-friendly; there are alternatives better suited to humans. However, this drawback is not very significant because ultimately users will be developing their own ontologies using authoring tools, or more generally, ontology development tools, instead of writing them directly in OWL.

A formal semantics describes the meaning of knowledge precisely. Precisely here means that the semantics does not refer to subjective intuitions, nor is it open to different interpretations by different people (or machines). The importance of a formal semantics is well-established in the domain of mathematical logic, for instance. One use of a formal semantics is to allow people to reason about the knowledge. For ontological knowledge, we may reason about the following:

- *Class membership.* If x is an instance of a class C , and C is a subclass of D , then we can infer that x is an instance of D .
- *Equivalence of classes.* If class A is equivalent to class B , and class B is equivalent to class C , then A is equivalent to C , too.
- *Consistency.* Suppose we have declared x to be an instance of the class A and that A is a subclass of $B \cap C$, A is a subclass of D , and B and D are disjoint. Then we have an inconsistency because A should be empty but has the instance x . This is an indication of an error in the ontology.
- *Classification.* If we have declared that certain property-value pairs are a sufficient condition for membership in a class A , then if an individual x satisfies such conditions, we can conclude that x must be an instance of A . [1]

4.5.2. Limitations of RDF

The expressivity of RDF and RDF Schema is deliberately very limited. RDF is limited to binary ground predicates, and RDF Schema is limited to a subclass hierarchy and a property hierarchy, with domain and range definitions of these properties (see also section 4.3 Resource Description Framework (RDF)).

RDF and RDFS allow the representation of *some* ontological knowledge. The main modeling primitives of RDF/RDFS concern the organization of vocabularies in typed hierarchies: subclass and subproperty relationships, domain and range restrictions, and instances of classes. However, a number of other features are missing like:

- **Local scope of properties.** `rdfs:range` defines the range of a property. In RDF Schema we cannot declare range restrictions that apply to some classes only. For example, we cannot say that cows eat only plants, while other animals may eat meat, too.
- **Disjointness of classes.** Sometimes we wish to say that classes are disjoint. For example, `male` and `female` are disjoint. But in RDF Schema we can only state subclass relationships, e.g., `female` is a subclass of `person`.

Putting it all together

- **Boolean combinations of classes.** Sometimes we wish to build new classes by combining other classes using union, intersection, and complement. For example, we may wish to define the class `person` to be the disjoint union of the classes `male` and `female`. RDF Schema does not allow such definitions.
- **Cardinality restrictions.** Sometimes we wish to place restrictions on how many distinct values a property may or must take. For example, we would like to say that a person has exactly two parents, or that a course is taught by at least one lecturer. Again, such restrictions are impossible to express in RDF Schema.
- **Special characteristics of properties.** Sometimes it is useful to say that a property is transitive (like “greater than”), unique (like “is mother of”), or the inverse of another property (like “eats” and “is eaten by”).

In Figure 20. Key purpose of major ontology languages, the main purpose of the ontology languages presented so far is depicted, showing in a simple diagram the restrictions posed by each technology as related towards the implementation of the “true” Semantic Web.

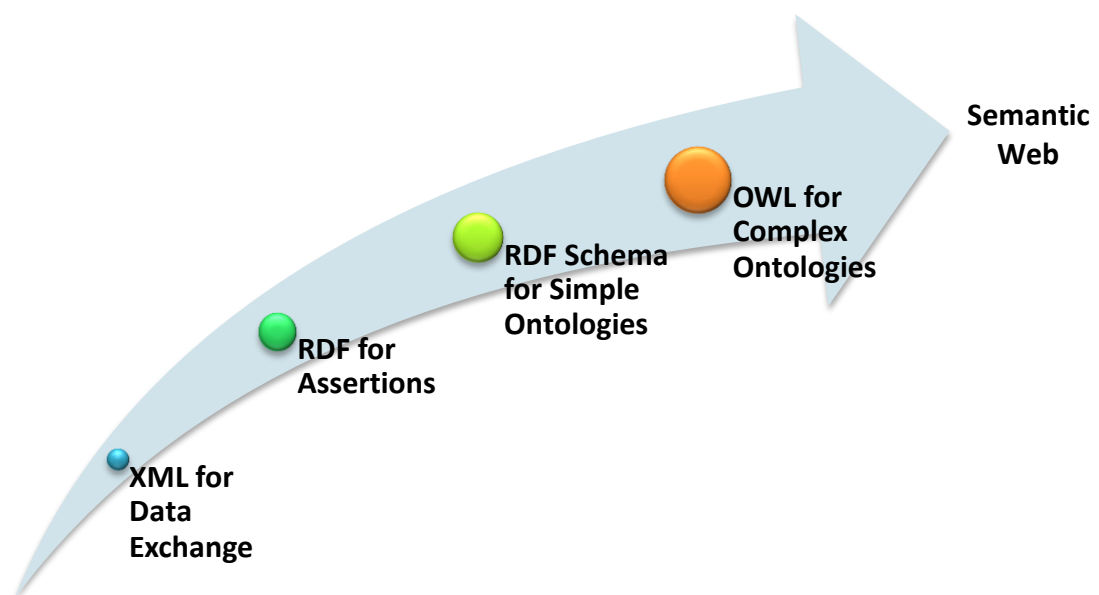


Figure 20. Key purpose of major ontology languages

Thus we need an ontology language that is richer than RDF Schema, a language that offers these features and more and this language is OWL, for the moment. [16]

4.5.3. The three versions of OWL

The full set of requirements for an ontology language (efficient reasoning support and convenience of expression for a language as powerful as a combination of RDF Schema with a full logic) seem unobtainable.

For this reason, these requirements have prompted W3C's Web Ontology Working Group to define OWL as three different sublanguages, each geared toward fulfilling different aspects of this full set of requirements. Each version encompasses the characteristics of another version to provide greater flexibility as is depicted in Figure 21. The 3 versions of OWL.

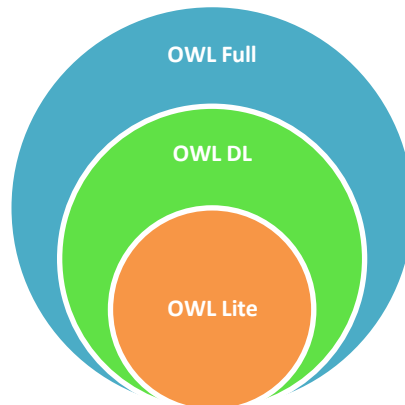


Figure 21. The 3 versions of OWL

There are strict notions of upward compatibility between these three sublanguages:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL Full

The entire language is called OWL Full and uses all the OWL language primitives. It also allows the combination of these primitives in arbitrary ways with RDF and RDF Schema. This includes the possibility (also present in RDF) of changing the meaning of the predefined (RDF or OWL) primitives by applying the language primitives to each other. For example, in OWL Full, we could impose a cardinality constraint on the class of all classes, essentially limiting the number of classes that can be described in any ontology.

The advantage of OWL Full is that it is fully upward-compatible with RDF, both syntactically and semantically: any legal RDF document is also a legal OWL Full document, and any valid RDF/RDF Schema conclusion is also a valid OWL Full conclusion. The disadvantage of OWL Full is that the language has become so powerful as to be undecidable, dashing any hope of complete (or efficient) reasoning support.

OWL DL (*Description Logic*)

Putting it all together

In order to regain computational efficiency, OWL DL is a sublanguage of OWL Full that restricts how the constructors from OWL and RDF may be used. Essentially application of OWL's constructors to each other is disallowed, thus ensuring that the language corresponds to a well-studied description logic.

The advantage of this is that it permits efficient reasoning support. The disadvantage is that we lose full compatibility with RDF. An RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document. Every legal OWL DL document is a legal RDF document.

OWL Lite

An even further restriction limits OWL DL to a subset of the language constructors. For example, OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality.

The advantage of this is a language that is both easier to grasp (for users) and easier to implement (for tool builders). The disadvantage is, of course, a restricted expressivity.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more expressive constructs provided by OWL DL. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the metamodeling facilities of RDF Schema (e.g., defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable because complete OWL Full implementations will be impossible. [1]

4.5.4. Brief description of the language

OWL builds on RDF and RDF Schema and uses RDF's XML-based syntax. Since this is the primary syntax for OWL, we use it here, but RDF/XML does not provide a very readable syntax. Because of this, other syntactic forms for OWL have also been defined:

- An XML-based syntax⁵ that does not follow the RDF conventions and is thus more easily read by human users.
- An abstract syntax, used in the language specification document⁵, that is much more compact and readable than either the XML syntax or the RDF/XML syntax.

⁵ Defined in <http://www.w3.org/TR/owl-xmlsyntax/>

Putting it all together

- A graphic syntax based on the conventions of UML (Unified Modeling Language), which is widely used and is thus an easy way for people to become familiar with OWL.

OWL documents are usually called *OWL ontologies* and are RDF documents. The root element of an OWL ontology is an `rdf:RDF` element, which also specifies a number of namespaces:

```
<rdf:RDF
  xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
```

An OWL ontology may start (**header**) with a collection of assertions for housekeeping purposes. These assertions are grouped under an `owl:Ontology` element, which contains comments, version control, and inclusion of other ontologies. `Owl:imports`, lists other ontologies whose content is assumed to be part of the current ontology.

Class element are defined using an `owl:Class` element. For example, we can define a class `associateProfessor` as follows:

```
<owl:Class rdf:ID="associateProfessor">
  <rdfs:subClassOf rdf:resource="#academicStaffMember"/>
</owl:Class>
```

Equivalence of classes can be defined using an `owl:equivalentClass` element:

```
<owl:Class rdf:ID="faculty">
  <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

There are two predefined classes in OWL, `owl:Thing` and `owl:Nothing`.

As far as **Property elements** are concerned, there are two kinds in OWL:

- Object properties, which relate objects to other objects. Examples are `isTaughtBy` and `supervises`.
- Data type properties, which relate objects to data type values. Examples are `phone`, `title`, and `age`. OWL does not have any predefined data types, nor does

Putting it all together

it provide special definition facilities. Instead, it allows one to use XML Schema data types, thus making use of the layered architecture of the Semantic Web.

Equivalence of properties can be defined through the use of the element `owl:equivalentProperty`.

We can enforce **restrictions** on Properties as follows. With `rdfs:subClassOf` we can specify a class *C* to be subclass of another class *C'*; then every instance of *C* is also an instance of *C'*.

Suppose we wish to declare instead that the class *C* satisfies certain conditions, that is, all instances of *C* satisfy the conditions. This is equivalent to saying that *C* is subclass of a class *C'*, where *C'* collects all objects that satisfy the conditions. That is exactly how it is done in OWL. Note that, in general, *C'* can remain anonymous.

`owl:allValuesFrom` is used to specify the class of possible values the property specified by `owl:onProperty` can take (in other words, all values of the property must come from this class).

`owl:hasValue` states a specific value that the property specified by `owl:onProperty` must have.

In general, an `owl:Restriction` element contains an `owl:onProperty` element and one or more restriction declarations. One type of restriction declaration defines restrictions on the kinds of values the property can take: `owl:allValuesFrom`, `owl:hasValue`, and `owl:someValuesFrom`. We conclude by noting that `owl:Restriction` defines an anonymous class which has no ID, is not defined by `owl:Class`, and has only local scope: it can only be used in the one place where the restriction appears. When we talk about classes, we must keep in mind the twofold meaning: classes defined by `owl:Class` with an ID, and local anonymous classes as collections of objects that satisfy certain restriction conditions, or as combinations of other classes. The latter are sometimes called *class expressions*.

Some **special properties** of property elements can be defined directly:

- `owl:TransitiveProperty` defines a transitive property, such as “has better grade than”, “is taller than”, or “is ancestor of”.
- `owl:SymmetricProperty` defines a symmetric property, such as “has same grade as” or “is sibling of”.

Putting it all together

- `owl:FunctionalProperty` defines a property that has at most one value for each object, such as “age”, “height”, or “directSupervisor”.
- `owl:InverseFunctionalProperty` defines a property for which two different objects cannot have the same value, for example, the property “isTheSocialSecurityNumberfor” (a social security number is assigned to one person only).

It is possible to talk about **Boolean combinations** (union, intersection, complement) of classes by using `owl:disjointWith`, `owl:unionOf`, `owl:intersectionOf`.

Instances of classes are declared as in RDF, like for example:

```
<rdf:Description rdf:ID="949352">
  <rdf:type rdf:resource="#academicStaffMember"/>
</rdf:Description>
```

or equivalently

```
<academicStaffMember rdf:ID="949352"/>
```

Unlike typical database systems, OWL does not adopt the *unique-names assumption*; just because two instances have a different name or ID does not imply that they are different individuals.

Although XML Schema provides a mechanism to construct user-defined data types (e.g., the data type of `adultAge` as all integers greater than 18, or the data type of all strings starting with a number), such derived data types cannot be used in OWL. In fact, not even all of the many built-in XML Schema data types can be used in OWL. The OWL reference document lists all the XML Schema data types that can be used, but these include the most frequently used types such as string, integer, Boolean, time, and date. [1]

4.6. Some well-known initiatives

4.6.1. Dublin Core Metadata Initiative (DCMI)

The Dublin Core metadata terms are a set of vocabulary terms which can be used to describe



resources for the purposes of discovery. The terms can be used to describe a full range of web resources: video, images, web pages etc. and physical resources such as books and objects like artworks. Dublin Core can be used for multiple purposes, from simple resource description, to combining metadata vocabularies of different metadata standards, to providing interoperability for metadata vocabularies in the linked data cloud and semantic web implementations. [2]

The mission of the Dublin Core Metadata Initiative is to provide simple standards to facilitate the finding, sharing and management of information. DCMI does this by:

- Developing and maintaining international standards for describing resources.
- Supporting a worldwide community of users and developers.
- Promoting widespread use of Dublin Core solutions.

The Simple Dublin Core Metadata Element Set (DCMES) consists of the following metadata elements: [27]

No	Element
1	Title
2	Creator
3	Subject
4	Description
5	Publisher
6	Contributor
7	Date
8	Type
9	Format
10	Identifier
11	Source
12	Language
13	Relation
14	Coverage
15	Rights

Putting it all together

Subsequent to the specification of the original 15 elements, an ongoing process to develop exemplary terms extending or refining the Dublin Core Metadata Element Set (DCMES) was begun. The additional terms were identified, generally in working groups of the Dublin Core Metadata Initiative⁶, and judged by the DCMI Usage Board to be in conformance with principles of good practice for the qualification of Dublin Core metadata elements.

The four currently approved DCMI namespace URIs are the following: [28]

URI	Description
http://purl.org/dc/terms/	All DCMI properties, classes and encoding schemes
http://purl.org/dc/dcmitype/	Classes in the DCMI Type Vocabulary
http://purl.org/dc/dcam/	Terms used in the DCMI Abstract Model
http://purl.org/dc/elements/1.1/	The Dublin Core Metadata Element Set, Version 1.1 (original 15 elements)

4.6.2. The Friend of a Friend (FOAF) project

FOAF is a project⁷ devoted to linking people and information using the Web. Regardless of whether information is in people's heads, in physical or digital documents, or in the form of factual data, it can be linked. FOAF integrates three kinds of network:



- *social networks* of human collaboration, friendship and association;
- *representational networks* that describe a simplified view of a cartoon universe in factual terms, and
- *information networks* that use Web-based linking to share independently published descriptions of this inter-connected world.

FOAF does not compete with socially-oriented Web sites; rather it provides an approach in which different sites can tell different parts of the larger story, and by which users can retain some control over their information in a non-proprietary format. FOAF is a descriptive vocabulary expressed using the Resource Description Framework (RDF). [29]

FOAF describes the world using simple ideas inspired by the Web. In FOAF descriptions, there are only various kinds of things and links, which are called

⁶ <http://www.dublincore.org/>

⁷ <http://www.foaf-project.org/>

Putting it all together

properties. The types of the things in FOAF are called *classes*. FOAF is therefore defined as a dictionary of terms, each of which is either a *class* or a *property*. Other projects alongside FOAF provide other sets of classes and properties, many of which are linked with those defined in FOAF.

FOAF descriptions are themselves published as linked documents in the Web (e.g. using RDF/XML or RDFa syntax). The result of the FOAF project is a network of documents describing a network of people. Each FOAF document is itself an encoding of a descriptive network structure. Although these documents do not always agree or tell the truth, they have the useful characteristic that they can be easily merged, allowing partial and decentralised descriptions to be combined in interesting ways.

FOAF collects a variety of terms; some describe people, some groups, some documents. Different kinds of application can use or ignore different parts of FOAF.

The main FOAF terms are grouped in the following broad categories:

- **Core-** These classes and properties form the core of FOAF. They describe characteristics of people and social groups that are independent of time and technology; as such they can be used to describe basic information about people in present day, historical, cultural heritage and digital library contexts. In addition to various characteristics of people, FOAF defines classes for Project, Organization and Group as other kinds of agent.
- **Social Web-** in addition to the FOAF core terms, there are a number of terms for use when describing Internet accounts, address books and other Web-based activities.

FOAF Core	Social Web
Agent	nick
Person	mbox
name	homepage
title	weblog
img	openid
depiction (depicts)	jabberID
familyName	mbox_shalsum
givenName	interest
knows	topic_interest
based_near	topic (page)
age	workplaceHomepage
made (maker)	workInfoHomepage
primaryTopic (primaryTopicOf)	schoolHomepage

Putting it all together

Project	publications
Organization	currentProject
Group	pastProject
member	account
Document	OnlineAccount
Image	accountName
	accountServiceHomepage
	PersonalProfileDocument
	tipjar
	sha1
	thumbnail
	logo

The namespace URI for the FOAF vocabulary is

`http://xmlns.com/foaf/0.1/`

Agreement between the FOAF Project and the Dublin Core Metadata Initiative

The FOAF Vocabulary and DCMI Metadata Terms are often used together in applications and both are consistently listed among the top vocabularies in the Linked Data space. As organizations, DCMI and the FOAF Project share a common interest in improving resource discovery across the boundaries of information silos on the Web. They also share a common concern for balancing centralization and decentralization by encouraging the stabilization of third-party extensions and companion vocabularies that enhance the usefulness of the vocabularies they maintain.

This led to an agreement in 2011 for specific measures to be undertaken in cooperation between DCMI and the FOAF Project -- measures aimed primarily at reinforcing the long-term viability of the FOAF vocabulary by addressing the risks inherent with having a single point of failure. The two organizations also see this cooperation as an opportunity for better integrating their vocabularies with alignments -- mutually declared mappings between semantically overlapping terms -- and for promoting the documentation of best-practice usage patterns in which the two vocabularies are used in combination.

Both organizations believe that arrangements of mutual support and cooperation among vocabulary maintainers can improve the long-term viability of RDF vocabularies in all niches of the Semantic Web ecosystem -- from vocabularies maintained by small, agile, time-limited projects or grass-roots initiatives to vocabularies maintained by stable cultural memory. [30]

4.6.3. DBpedia

DBpedia⁸ is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows a user to ask sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data. This makes it easier for the amazing amount of information in Wikipedia to be used in new and interesting ways, and inspire new mechanisms for navigating, linking and improving the encyclopedia itself. [3]



The DBpedia knowledge base currently describes more than 3.64 million things, out of which 1.83 million are classified in a consistent Ontology. The dataset consists of 1 billion pieces of information (RDF triples) out of which 385 million were extracted from the English edition of Wikipedia and roughly 665 million were extracted from other language editions and links to external datasets.

DBpedia has grown into one of the central knowledge sources of mankind and is maintained by thousands of contributors. Wikipedia articles consist mostly of free text, but also contain different types of structured information, such as infobox templates, categorization information, images, geo-coordinates, and links to external Web pages. For instance, Figure 22. Example of a Wikipedia Infobox, shows the source code and the visualization of an infobox template containing structured information about the town of Innsbruck. DBpedia uses the Resource Description Framework (RDF) as a flexible data model for representing extracted information and for publishing it on the Web. [31]

The namespace URI for DBpedia ontology is

<http://dbpedia.org/ontology/>

⁸ <http://dbpedia.org/About>

```

{{Infobox Town AT |
    name = Innsbruck |
    image_coa = InnsbruckWappen.png |
    image_map = Karte-tirol-I.png |
    state = [[Tyrol]] |
    regbzkr = [[Statutory city]] |
    population = 117,342 |
    population_as_of = 2006 |
    pop_dens = 1,119 |
    area = 104.91 |
    elevation = 574 |
    lat_deg = 47 |
    lat_min = 16 |
    lat_hem = N |
    lon_deg = 11 |
    lon_min = 23 |
    lon_hem = E |
    postal_code = 6010-6080 |
    area_code = 0512 |
    licence = I |
    mayor = Hilde Zach |
    website = [http://innsbruck.at] |
}}

```

Innsbruck	
	
Country	 Austria
State	Tyrol
Administrative region	Statutory city
Population	117,342 (2006)
Area	104.91 km ²
Population density	1,119 /km ²
Elevation	574 m
Coordinates	47°16′N 11°23′E﻿•﻿47°16′N 11°23′E
Postal code	6010-6080
Area code	0512
Licence plate code	I
Mayor	Hilde Zach
Website	www.innsbruck.at

Figure 22. Example of a Wikipedia Infobox

Each thing in the DBpedia data set is identified by a URI reference of the form `http://dbpedia.org/page/Name`, where `Name` is taken from the URL of the source Wikipedia article, which has the form `http://en.wikipedia.org/wiki/Name`. Thus, each resource is tied directly to an English-language Wikipedia article. To illustrate this let us use the following example. If we search the term “Venizelos” in Wikipedia we get the results page shown in Figure 23. Example of a Wikipedia results page.



The screenshot shows the Wikipedia article for "Venizelos". At the top, there is a navigation bar with "Article", "Discussion", "Read", "Edit", and "View history" tabs. Below this is a search bar. The main content area features a large "W" logo and a message: "Thank you for protecting Wikipedia. (We're not done yet.)". The article title "Venizelos" is prominently displayed, followed by the text "From Wikipedia, the free encyclopedia". The article content begins with "Venizelos (Greek: Βενζέλος) is a Greek surname; it may refer to:" followed by a bulleted list of names and dates. Below this is a "See also" section with a bulleted list of links. At the bottom, there is a note about internal links and a small icon.

Figure 23. Example of a Wikipedia results page

Putting it all together

The corresponding page of DBpedia is shown in Figure 24. Corresponding DBpedia page of a Wikipedia resource.



The screenshot shows the DBpedia page for 'Venizelos'. The page title is 'About: Venizelos'. Below the title, it states 'An Entity of Type : [Thing](#), from Named Graph : <http://dbpedia.org>, within Data Space : dbpedia.org'. The page content includes a table of properties and values, a list of related entities, and a list of related topics. The table of properties and values is as follows:

Property	Value
dbpedia-owl:abstract	<ul style="list-style-type: none"> Venizelos (Βενιζέλος) es un apellido griego. Puede referirse a: Eleftherios Venizelos (1864-1936), político y primer ministro griego; Sofoklis Venizelos (1894-1964), político y primer ministro griego, hijo de Eleftherios. Evangelos Venizelos (n. 1957), político griego, no relacionado con los anteriores. Βενιζέλος — греческая фамилия.
dbpedia-owl:wikiPageDisambiguate	<ul style="list-style-type: none"> dbpedia:Eleftherios_Venizelos dbpedia:Evangelos_Venizelos dbpedia:Athens_International_Airport dbpedia:Eleftherios_Venizelos_Crete dbpedia:Sofoklis_Venizelos
dcterms:subject	<ul style="list-style-type: none"> category:Greek-language_surnames
rdfs:comment	<ul style="list-style-type: none"> Venizelos (Βενιζέλος) es un apellido griego. Puede referirse a: Eleftherios Venizelos (1864-1936), político y primer ministro griego; Sofoklis Venizelos (1894-1964), político y primer ministro griego, hijo de Eleftherios. Evangelos Venizelos (n. 1957), político griego, no relacionado con los anteriores. Βενιζέλος — греческая фамилия.
rdfs:label	<ul style="list-style-type: none"> Venizelos Venizelos Venizélos Venizelos Venizélos Βενιζέλος
foaf:page	<ul style="list-style-type: none"> http://en.wikipedia.org/wiki/Venizelos
is owl:sameAs of	<ul style="list-style-type: none"> http://www4.wiwiwss.fu-berlin.de/flickrwrappr/photos/Venizelos
is foaf:primaryTopic of	<ul style="list-style-type: none"> http://en.wikipedia.org/wiki/Venizelos

Below the table, there is a section for 'Browse using:' with links to various tools: [OpenLink Data Explorer](#), [Zitgist Data Viewer](#), [Marbles](#), [DISCO](#), [Tabulator](#), [Raw Data in: CSV](#), [RDF \(N-Triples\)](#), [N3/Turtle](#), [JSON XML](#), [OData \(Atom JSON\)](#), [Microdata \(JSON HTML\)](#), [JSON-LD](#), and [About](#). There are also logos for 'POWERED BY VIRTUOSO', 'LINKING OPENDATA', 'W3C SPARQL', 'OPEN DATA', and 'W3C XHTML + RDFa'. At the bottom, it states 'This content was extracted from Wikipedia and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)'.

Figure 24. Corresponding DBpedia page of a Wikipedia resource

The main components of DBpedia-Live system are as follows:

- Local Wikipedia: The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) which enables an application to get a continuous stream of updates from a wiki. OAI-PMH is also used to feed updates into DBpedia-Live Extraction Manager.
- Mapping Wiki: DBpedia mappings can be found at <http://mappings.dbpedia.org>. It is also a wiki. We can also use OAI-PMH to get stream of updates in DBpedia mappings. Basically, a change of mapping affects several Wikipedia pages, which should be reprocessed.
- DBpedia-Live Extraction Manager: This component is the actual DBpedia-Live extraction framework. When there is a page that should be processed, the framework applies the extractors on it. After processing a page, the newly extracted triples are inserted into the backend triple store (Virtuoso),

Putting it all together

overwriting the old triples. The newly extracted triples are also written as N-Triples file and compressed. Other applications or DBpedia-Live mirrors that should always be in synchronization with our DBpedia-Live can download those files and feed them into its own triplestore.

This architecture is depicted in Figure 25. DBpedia Live-System Architecture. [32]

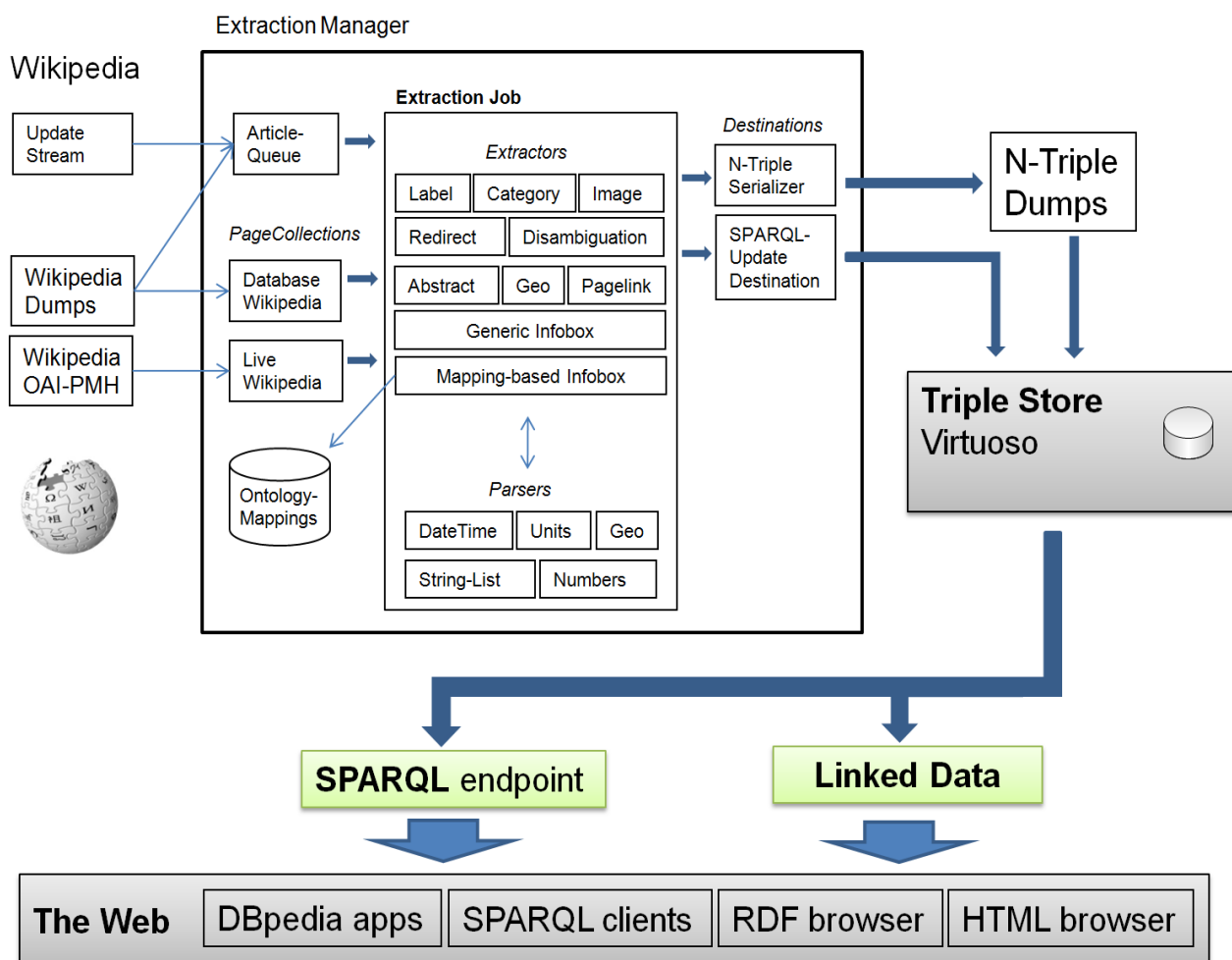


Figure 25. DBpedia Live-System Architecture (source: [32])

To explore the DBpedia database and understand the way that resources are interconnected, we can use a tool developed by DBpedia, *DBpedia Relationship Finder*. With this tool we can visualize the relationship between resources. In a simple form the user enters the resources he wishes to examine and makes some simple choices. On the results page the relationship between the resources is depicted.

As an example, the relationship of the resources Leipzig and University of Leipzig is shown in Figure 26. DBpedia Relationship Finder example. [33]

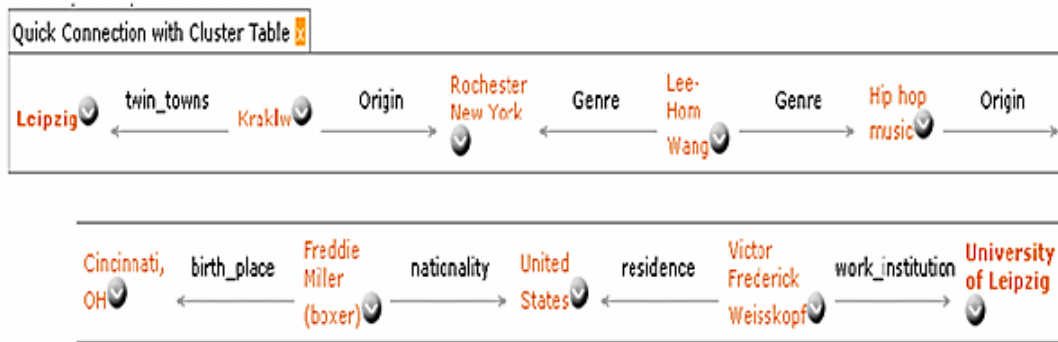


Figure 26. DBpedia Relationship Finder example

An advancement of *DBpedia Relationship Finder* is *RelFinder*. [34]

4.6.4. Simple Knowledge Organization System (SKOS)

Simple Knowledge Organization System (SKOS) is a family of formal languages designed for representation of thesauri, classification schemes, taxonomies, subject-heading systems, or any other type of structured controlled vocabulary. SKOS is



built upon RDF and RDFS, and its main objective is to enable easy publication of controlled structured vocabularies for the Semantic Web. SKOS is currently developed within the W3C framework.⁹ The name SKOS was chosen to emphasize the goal of providing a simple yet powerful framework for expressing knowledge organization systems in a machine-understandable way.

SKOS Core provides a model for expressing the basic structure and content of concept schemes. A 'concept scheme' is defined as a set of concepts, optionally including statements about semantic relationships between those concepts. Thesauri, classification schemes, subject heading lists, taxonomies, 'folksonomies', and other types of controlled vocabulary are all examples of concept schemes. Concept schemes are also embedded in glossaries and terminologies. [35]

The namespace URI for the SKOS vocabulary is

<http://www.w3.org/2004/02/skos/core#>

The SKOS vocabulary is the following and can be accessed with the use of the above mentioned URI (usually abbreviated with the prefix *skos*) followed by the suitable term. [36]

⁹ <http://www.w3.org/2004/02/skos/>

URI
<code>skos:Concept</code>
<code>skos:ConceptScheme</code>
<code>skos:inScheme</code>
<code>skos:hasTopConcept</code>
<code>skos:topConceptOf</code>
<code>skos:altLabel</code>
<code>skos:hiddenLabel</code>
<code>skos:prefLabel</code>
<code>skos:notation</code>
<code>skos:changeNote</code>
<code>skos:definition</code>
<code>skos:editorialNote</code>
<code>skos:example</code>
<code>skos:historyNote</code>
<code>skos:note</code>
<code>skos:scopeNote</code>
<code>skos:broader</code>
<code>skos:broaderTransitive</code>
<code>skos:narrower</code>
<code>skos:narrowerTransitive</code>
<code>skos:related</code>
<code>skos:semanticRelation</code>
<code>skos:Collection</code>
<code>skos:OrderedCollection</code>
<code>skos:member</code>
<code>skos:memberList</code>
<code>skos:broadMatch</code>
<code>skos:closeMatch</code>
<code>skos:exactMatch</code>
<code>skos:mappingRelation</code>
<code>skos:narrowMatch</code>
<code>skos:relatedMatch</code>

An example graph using some of the above terms of the SKOS vocabulary is depicted in Figure 27. Example of RDF graph using the SKOS Core Vocabulary (taken from the UK Archival Thesaurus (UKAT)). [37]

Putting it all together

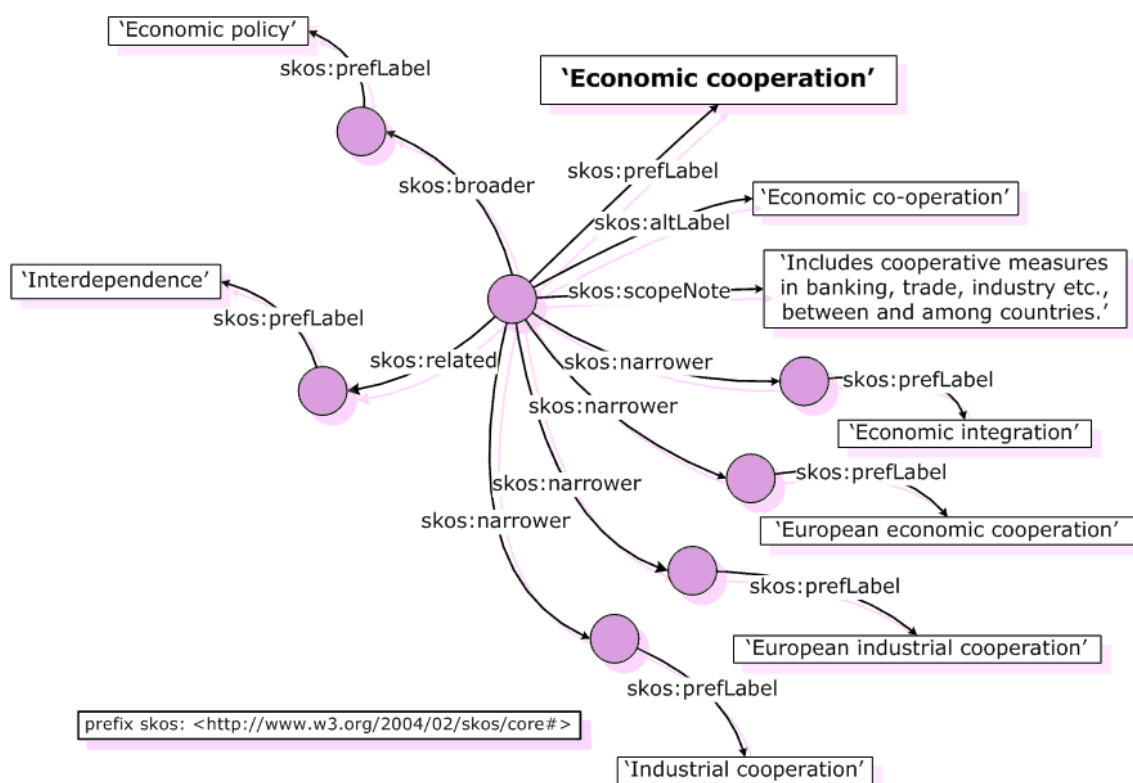


Figure 27. Example of RDF graph using the SKOS Core Vocabulary (source: [37])

4.7. SPARQL

...I nodded, to show that I followed his reasoning.

“It is very customary for pawnbrokers in England, when they take a watch, to scratch the number of the ticket with a pin-point upon the inside of the case.



It is more handy than a label, as there is no risk of the number being lost or transposed.

There are no less than four such numbers visible to my lens on the inside of this case. Inference, —that your brother was often at low water. Secondary inference,—that he had occasional bursts of prosperity, or he could not have redeemed the pledge. Finally, I ask you to look at the inner plate, which contains the key-hole.

Look at the thousands of scratches all round the hole,—marks where the key has slipped. What sober man’s key could have scored those grooves? But you will never see a drunkard’s watch without them. He winds it at night, and he leaves these traces of his unsteady hand. Where is the mystery in all this?”

“It is as clear as daylight,” I answered. “I regret the injustice which I did you. I should have had more faith in your marvelous faculty.”

Sir Arthur Conan Doyle: THE SIGN OF FOUR (chapter one- The Science of Deduction)

After managing to store data in ways that are convenient for humans, the big question is how to mine the information from this ocean of data. More and more people are using the query language SPARQL (pronounced “sparkle”) to pull data from either public or private data. Whether this data is part of a semantic web project or an integration of two inventory databases on different platforms behind the same firewall, SPARQL is making it easier to access it.

“Trying to use the Semantic Web without SPARQL is like trying to use a relational database without SQL.”

Tim Berners-Lee, W3C Director and Web inventor

But first, what is SPARQL? The name is a recursive acronym for SPARQL Protocol and RDF Query Language, which is described by a set of specifications from the W3C.



SPARQL was not designed to query relational data, but to query data conforming to the RDF data model. The “RQL” part of its name shows that SPARQL is designed to query RDF, but you’re not limited to querying data stored in one of the RDF formats. Commercial and open source utilities are available to treat relational data, XML, spreadsheets, and other formats as RDF so that you can issue SPARQL queries against these data sources — or against combinations of these sources, which is one of the most powerful aspects of the SPARQL/RDF combination.

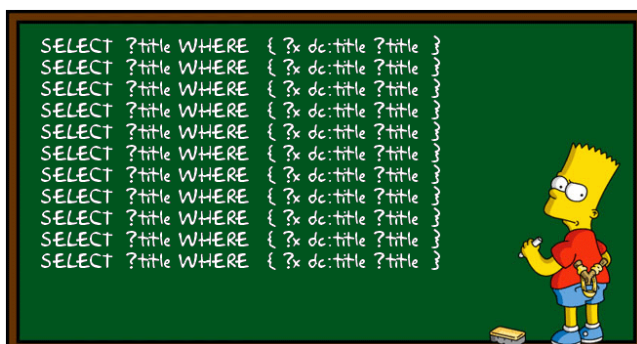
The “Protocol” part of SPARQL’s name refers to the rules for how a client program and a SPARQL processing server exchange SPARQL queries and results. These rules are specified in a separate document from the query specification document and are mostly an issue for SPARQL processor developers. You can go far with the query language without worrying about the protocol, so this book doesn’t go into any detail about it.

To sum up, we use a Query Language

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?website
FROM    <http://planetrdf.com/bloggers.rdf>
WHERE { ?person foaf:weblog ?website ;
        foaf:name ?name .
        ?website a foaf:Document
      }
```

... and a Protocol:

```
http://.../qps?query-lang=http://www.w3.org/TR/rdf-
sparql-query/
```



in order to find names and websites of contributors to PlanetRDF.

Generally a SPARQL query says “I want these pieces of information from the subset of the data that

Putting it all together

meets these conditions.” We can describe the conditions with triple patterns, which are similar to RDF triples but may include variables to add flexibility in how they match against the data.

4.7.1. SPARQL query basic form

SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions, using a select-from-where syntax, like SQL does. It can also support extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs. SPARQL queries take the following general form, showing the sections into which a query may be broken down and the clause or keyword which defines that section. The whole point of SPARQL is to provide a formal language with which to ask meaning-driven questions. [4]

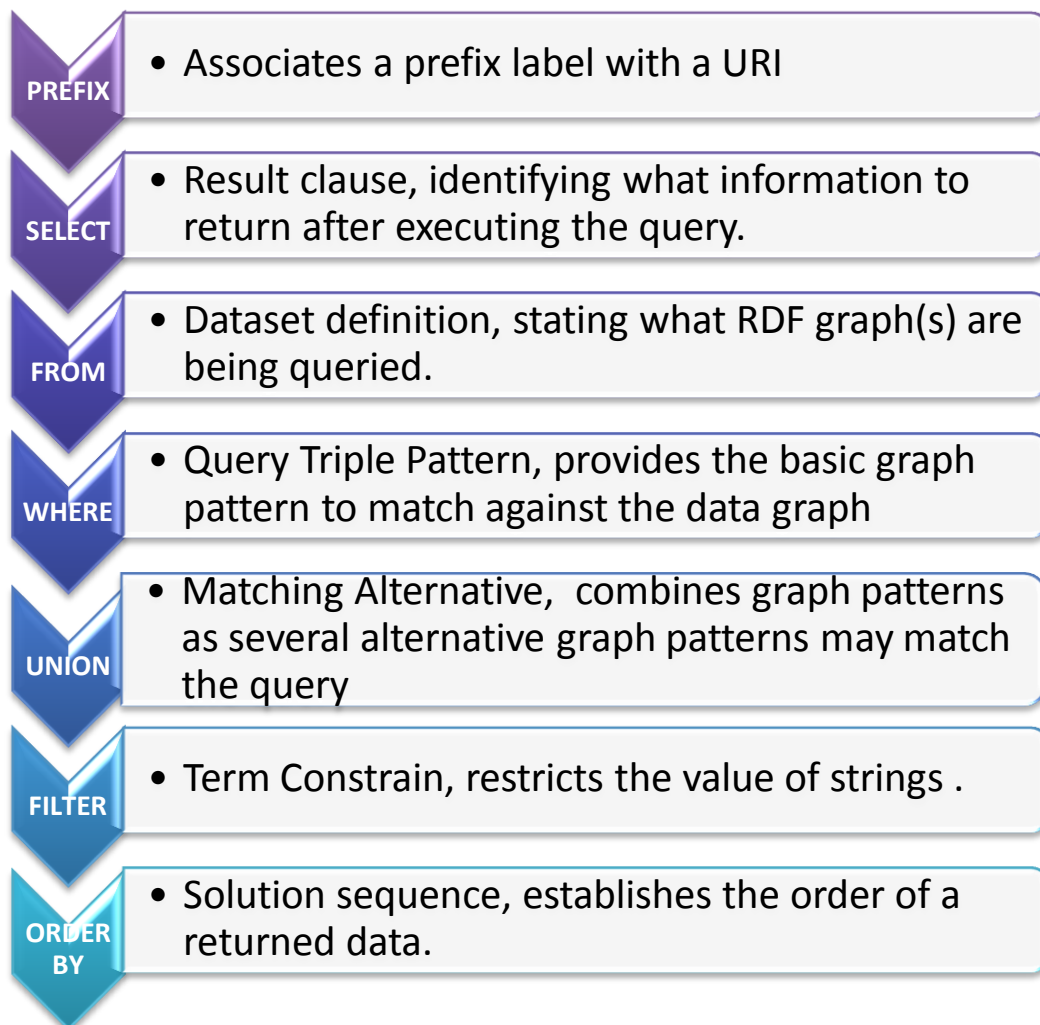


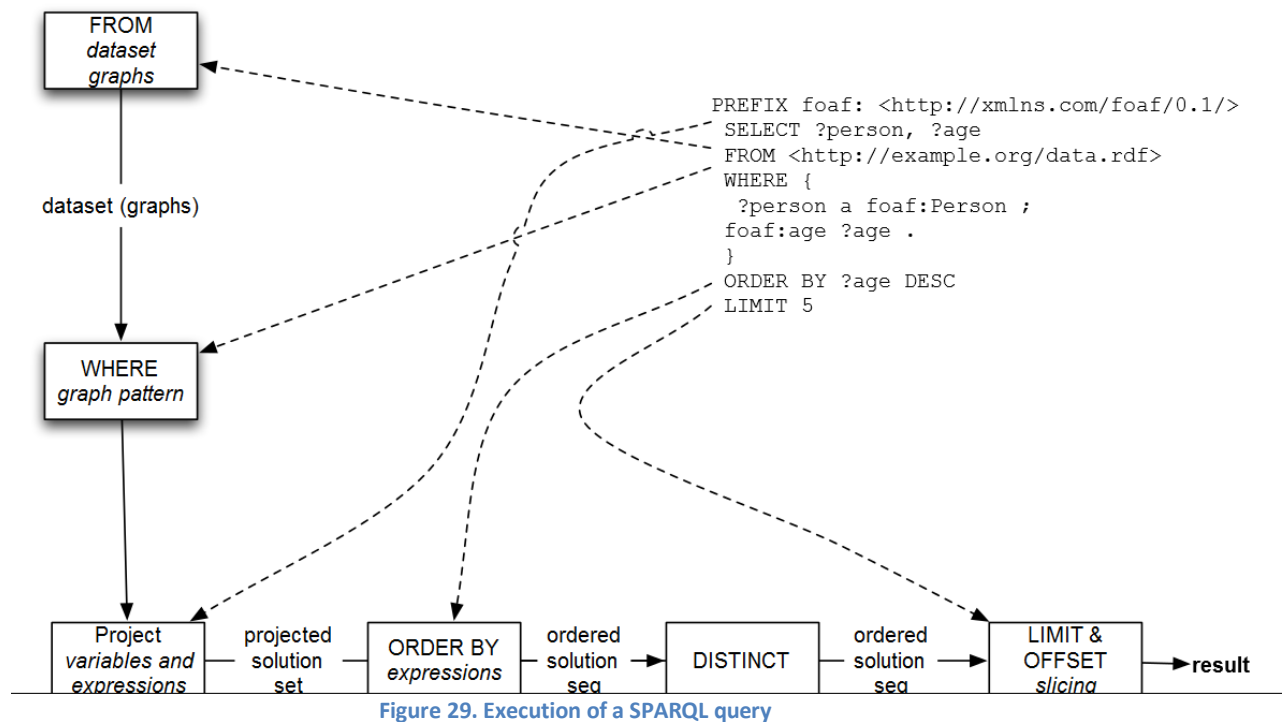
Figure 28. SPARQL query basic form

The basic principles in SPARQL queries are:

Putting it all together

- Make query look like the data with variables substituted (triple patterns).
- Generalise RDF triples to be 3-array of RDF Terms (URI, blank node, literal).
- SPARQL variables are prefixed with a “?”.
- In a query graph pattern, they match any node (resource or literal) in the RDF dataset.
- Triple patterns are just like triples, except that any of the parts of a triple can be replaced with a variable.
- The `SELECT` result clause returns a table of variables and values that satisfy the query. [38]

Figure 29. Execution of a SPARQL query shows in a graphical way how a SPARQL query is executed.



As we can see the SPARQL query's `WHERE` clause says “pull this data out of the dataset,” and the `SELECT` part names which parts of that pulled data you actually want to see. [39]

Putting it all together

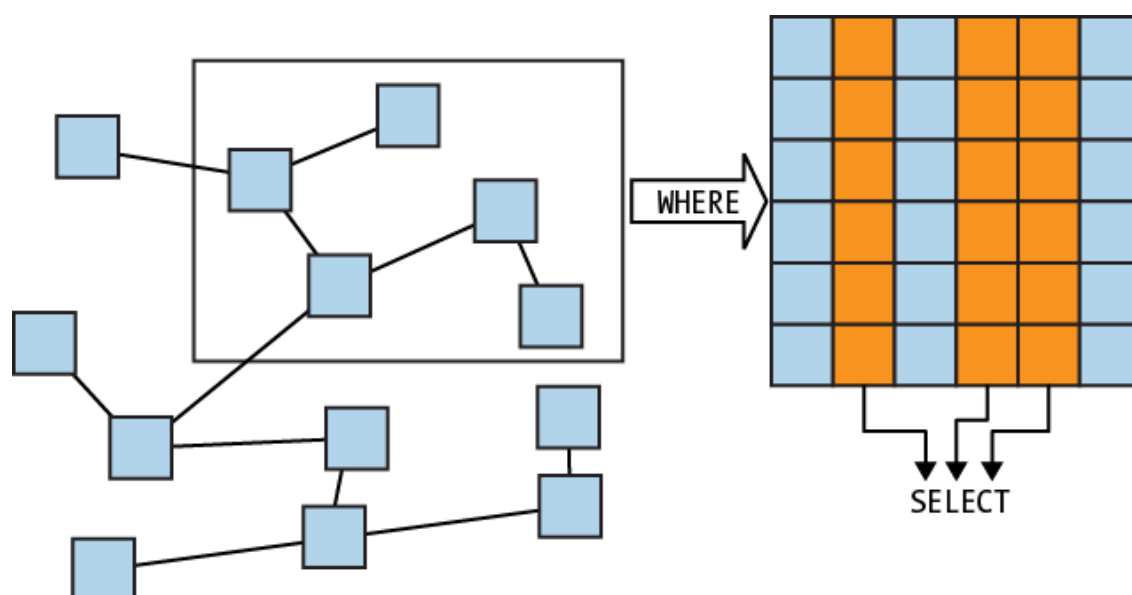


Figure 30. SPARQL design: WHERE specifies data to pull out; SELECT picks which data to display (source: [4])

SPARQL's syntax and query clauses will be further examined in section 7.2.3.2 Deep into SPARQL where their functionality will be analyzed in detail when considering the specific application of this thesis.

4.7.2. Querying a Public Data Source

In order to query public data sources, we need no special software, because these data collections are often made publicly available through a SPARQL endpoint. This is a web service that accepts SPARQL queries.

One of the most popular SPARQL endpoint of DBpedia is SNORQL. Like many SPARQL endpoints, DBpedia in order to make it very easy to explore its data, includes a web form in <http://dbpedia.org/snorql/>, where we can enter a query and then explore the results. DBpedia uses a program called SNORQL to accept these queries and return the answers on a web page.

SPARQL Explorer for <http://dbpedia.org/sparql>

SPARQL:

```

PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT * WHERE {
  ...
}
```

Results: Browse Go! Reset

Figure 31. SNORQL endpoint of DBpedia

Let us assume that we want DBpedia to return a list of albums produced by the great producer David Foster and the artists who made those albums.

The clue in this story is that if Wikipedia has a page for Some Topic at http://en.wikipedia.org/wiki/Some_Topic, the DBpedia URI to represent that resource is usually page http://dbpedia.org/page/Some_Topic. So after finding the Wikipedia page for the producer at http://en.wikipedia.org/wiki/David_Foster, we visit http://dbpedia.org/page/David_Foster where we found plenty of information and knew that this was the right URI to represent him in queries.

The producer and musicalArtist properties that we plan to use in our query are from the <http://dbpedia.org/ontology/> namespace, which is not declared on the SNORQL query input form, so we included a declaration for it in our query:

```

PREFIX d: <http://dbpedia.org/ontology/>
SELECT ?artist ?album
WHERE
{
  ?album d:producer :David_Foster .
  ?album d:musicalArtist ?artist .
}
```

This query pulls out triples about albums produced by David Foster and the artists listed for those albums and asks for the values that got bound to the `?artist` and

Putting it all together

?album variables. If we replace the default query on the SNORQL web page with this one and click the Go button, SNORQL displays the results underneath the query, as shown in Figure 32. SNORQL results page of a sample SPARQL query:

SPARQL results:	
artist	album
:Michael_Jackson	:Scream/Childhood
:Toni_Braxton	:Un-Break_My_Heart
:Monica_%28entertainer%29	:For_You_I_Will_%28Monica_song%29
:Andrea_Bocelli	:Somos_Novios_%28It%27s_Impossible%29
:Olivia_Newton-John	:Twist_of_Fate_%28Olivia_Newton-John_song%29
:Barbra_Streisand	:Tell_Him_%28Barbra_Streisand_and_Celine_Dion_song%29
:Celine_Dion	:Tell_Him_%28Barbra_Streisand_and_Celine_Dion_song%29
:Whitney_Houston	:I_Have_Nothing
:Destiny%27s_Child	:Stand_Up_for_Love
:Chicago_%28band%29	:If_She_Would_Have_Been_Faithful...
:Bryan_Adams	:I_Finally_Found_Someone
:Celine_Dion	:Because_You_Loved_Me
:Air_Supply	:The_One_That_You_Love
:Celine_Dion	:To_Love_You_More
:Celine_Dion	:Live_%28for_the_One_I_Love%29
:Celine_Dion	:In_Some_Small_Way
:Celine_Dion	:The_Power_of_the_Dream

Figure 32. SNORQL results page of a sample SPARQL query

This list of results is only the beginning of a much longer list, and even that may not be complete—remember, Wikipedia is maintained by volunteers, and while there are some quality assurance efforts in place, they are dwarfed by the scale of the data to work with. These artists and songs have their own Wikipedia pages and associated data, and the associated data includes more readable versions of the names that we can ask for in a query.

Another popular endpoint of DBpedia is Virtuoso. Virtuoso is the endpoint that will be used for the purpose of this thesis and will be presented later in section 7.2 Step 2. – Searching the Semantic Web.

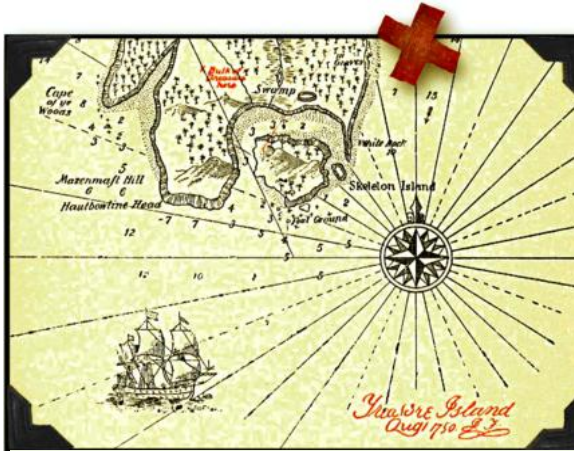
5. Roadmap

In the previous chapters information was given in brief to present the current situation regarding the World Wide Web and the problems that arise from the technologies used so far. The main of the problems is the inability of the machines to comprehend the behemoth of data and information that exists on the Web.

To this direction continuous scientific effort has produced new technologies aiming to establish reasoning by machines. Data available is transformed in a systematic way so as to remove ambiguity among humans and give mind capabilities to computers.

In the following chapters the above mentioned technologies will be used and examined in further detail along with existing ones to produce a tangible application.

6. Building our Application - Thesis



The paper had been sealed in several places with a thimble by way of seal; the very thimble, perhaps, that I had found in the captain's pocket. The doctor opened the seals with great care, and there fell out the map of an island, with latitude and longitude, soundings, names of hills and bays and inlets, and every particular that would be needed to bring a ship to a safe anchorage upon its shores. It was

about nine miles long and five across, shaped, you might say, like a fat dragon standing up, and had two fine land-locked harbours, and a hill in the center part marked "The Spy-glass." There were several additions of a later date, but above all, three crosses of red ink—two on the north part of the island, one in the southwest—and beside this last, in the same red ink, and in a small, neat hand, very different from the captain's tottery characters, these words: "Bulk of treasure here."

Over on the back the same hand had written this further information:

Tall tree, Spy-glass shoulder, bearing a point to the N. of N.N.E.

Skeleton Island E.S.E. and by E. Ten feet.

The bar silver is in the north cache; you can find it by the trend of the east hummock, ten fathoms south of the black crag with the face on it.

The arms are easy found, in the sand-hill, N.

point of north inlet cape, bearing E. and a quarter N. J.F.

TREASURE ISLAND by Robert Louis Stevenson

As mentioned in the introduction, the scope of the current thesis is to use the Semantic Web technologies to retrieve information regarding «worth to see» places that are in proximity to a user's position. In order to do so, the application that will be developed in the context of this thesis makes use, apart from Semantic Web technologies, Google maps, Google Latitude and Geocoding services.

Before diving into the sanctuary of the code, it is important to explore how a user's position is described upon earth.

6.1. Geocoding

According to Wikipedia **Geocoding** is the process of finding associated geographic coordinates (often expressed as latitude and longitude) from other geographic data, such as street addresses, or zip codes (postal codes). With geographic coordinates the features can be mapped and entered into Geographic Information Systems, or the coordinates can be embedded into media such as digital photographs via geotagging. Reverse geocoding is the opposite: finding an associated textual location such as a street address, from geographic coordinates. [40]

A simple method of geocoding is address interpolation (the method of constructing new data points within the range of a discrete set of known data points). This method makes use of data from a street geographic information system where the street network is already mapped within the geographic coordinate space. Each street segment is attributed with address ranges (e.g. house numbers from one segment to the next). Geocoding takes an address, matches it to a street and specific segment (such as a block, in towns that use the "block" convention). Geocoding then interpolates the position of the address, within the range along the segment.



In this process a geocoder - a piece of software or a (web) service- provide great help. Moreover Google uses geocoding in the Google Maps and has simplified the geocoding process by providing users with a geocoder that converts mailing addresses, city names and zip codes into usable latitude and longitude points. So with Google's Geocoding API, mobile and website developers can directly access a geocoder using an HTTP request.

6.2. Google Maps

According to Wikipedia Google Maps (formerly Google Local) is a web mapping service application and technology provided by Google, free (for non-commercial use), that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit, and maps embedded on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, car, bike (beta) or public transport and an urban business locator for numerous countries around the world. [41]

Putting it all together

All these began in 2005 when Google company expanded from searching to mapping. While at that time other online maps act more like static Web sites, Google maps were offering search capabilities with some very dynamic interface design. Not only that, but Google Maps were fitting perfectly into the browser window, and were acting more like an application, updating in real time as the user drag the map around and zoom in and out.

Over the years Google maps became a powerful tool in the programmers' hands. Today hundreds of sites and programs use the service as it is offering, according to Google maps supporting powerful, user-friendly mapping technology and local business information -- including business locations, contact information, and driving directions. [42]

More over the user can enjoy the following unique features:

- Integrated business search results – One can find business locations and contact information all in one location, integrated on the map.
- Draggable maps - Click and drag maps to view adjacent sections instantly (no long waits for new areas to download).
- Satellite imagery - View a satellite image (or a satellite image with superimposed map data) of your desired location that you can zoom and pan.
- Earth view - Click the Earth button to view 3D imagery and terrain from Google Earth on Maps that you can zoom, pan, and tilt.
- Street View - View and navigate within street-level imagery.
- Detailed directions – One can enter an address and let Google Maps plot the location and driving directions.



Google Maps provides high-resolution aerial or satellite images for most urban areas in the United States (including Hawaii, Alaska, Puerto Rico, and the U.S. Virgin Islands), Canada, and the United

Kingdom, as well as parts of Australia and many other countries. The high-resolution imagery has been used by Google Maps to cover all of Egypt's Nile Valley, Sahara desert and Sinai. Google Maps also covers many cities in the English speaking areas. However, Google Maps is not solely an English maps service, since its service is intended to cover the world. Google has blurred some areas for security.

Putting it all together

With the introduction of an easily pannable and searchable mapping and satellite imagery tool, Google's mapping engine prompted a surge of interest in satellite imagery. Sites were established which feature satellite images of interesting natural and man-made landmarks, including such novelties as "large type" writing visible in the imagery, as well as famous stadia and unique geological formations. Although Google uses the word satellite, most of the high-resolution imagery of cities is aerial photography taken from aircraft flying at 800–1500 feet rather than from satellites; while most of the rest of the imagery is in fact from satellites. Although the aerial views images are undated, they occasionally coincide with known events. For example, as of Oct. 8, 2011, the aerial view of the Hollywood region of Los Angeles shows the street-closures and temporary structures related to the 2011 Academy Awards ceremony.

7. Step By Step approach

A step by step approach to our software application as it is shown at the website www.worth2C.info, will provide a clear view to all the processes that are performed in the background.

7.1. Step 1. – Where is the user located?



As it was stated in the Introduction, the scope of the current thesis is to provide a user with the capability to use an Internet-capable device and with few simple commands to view, on a map, his position and all the interesting places around him, along with relevant information. In order to do so we use the Semantic Web technologies to retrieve information regarding worth to see places that are in proximity to the user's position. But first of all it is essential to identify his position. This is accomplished with Google Latitude API.

7.1.1. Google Latitude

Google Latitude [43] is a location-aware mobile application developed by Google. Latitude allows a mobile phone user to allow certain people to view their current location. Via their own Google Account, the user's cell phone location is mapped on Google Maps. The user can control the accuracy and details of what each of the other users can see — an exact location can be allowed, or it can be limited to identifying the city only. For privacy, it can also be turned off by the user, or a location can be manually entered. Users have to explicitly opt in to Latitude, and may only see the location of those friends who have decided to share their location with them.

Google Latitude is executed also on PC browsers and uses the Geolocation API. It has automated location detection on mobile phones using Cellular positioning, Wi-Fi Positioning, and GPS. This location determination is achieved by using XPS (hybrid positioning system), a technology developed by a company named Skyhook Wireless. [44]

XPS uses the known position of reference points and satellite signals to determine the location of mobile devices. As reference points, XPS uses Wi-Fi routers and Cellular base-stations (Cell ID), each of which broadcasts a signal that includes a unique identifier. When a device is in range of any of these reference points, Skyhook's system matches the signal data and unique identifiers to those in its database and also incorporates GPS satellite signals to determine the user's location. One data element used to determine location is the Received Signal Strength Indicator (RSSI), a

Putting it all together

measurement of the strength of an access point's signal at the time it is received by the mobile device. XPS uses the RSSI data to help determine how close a mobile device is to a router when it requests a location calculation and whether it is operating indoors or outdoors. [45]

To sum up, when a device, application, or service requests its location, in less than two seconds XPS:

- Scans the area to collect data on nearby Wi-Fi access points
- Connects to one or more GPS satellites
- Identifies nearby cellular towers
- Feeds all of this information to advanced hybrid positioning algorithms to determine location

To use Google Latitude, a user must have a Google account and access the service from the URL <http://m.google.com/latitude> , when using a mobile device or from the following URL when using a PC: <http://www.google.com/latitude>

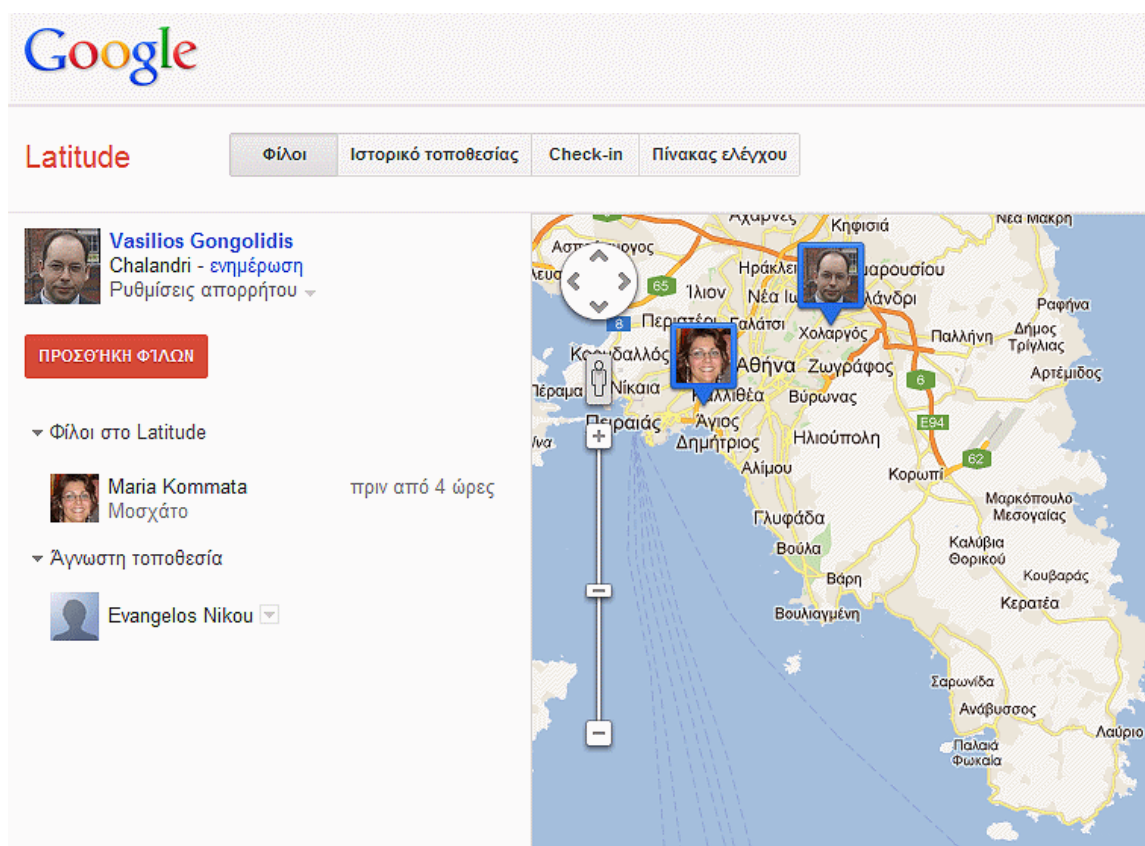


Figure 33. Google Latitude initial page

Putting it all together

7.1.2. Google Latitude Badge

According to the description provided for Google Latitude Badge [46], with this application you can share your Google Latitude location publicly on a blog or web site. You may choose to share only your city-level location or the best available one.

The user must take into consideration that when the Badge is enabled, his location will be visible to everyone, including people he has hid from using Google Latitude.

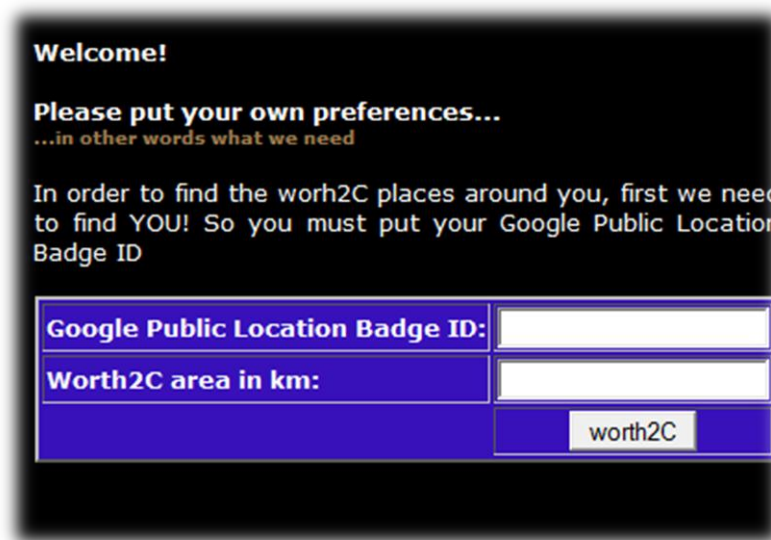
To identify his/her Google Latitude Badge, a user must:

1. Log on to Google Latitude
2. Select *Application Options*
3. Select *Activate and show best available location*

As soon as the user activates the Latitude Badge application, a user ID is given to him (as it is shown below), which contains all the information about his position. All the information can be revealed in XML or JSON format (see also section 4.3.3.3.5. JSON (JavaScript Object Notation)).

```
|<!--Google Public Location Badge -->
<iframe src="http://www.google.com/latitude/apps/badge
/api?user=-845643805XXXXXX&type=iframe&maptype=roadmap"
Width="180" height="300" frameborder="0"></ iframe>
```

For our application to work, the Badge ID obtained by the user must be obtained. Therefore in the index page of our site there are two input boxes: the first one for the badge ID and the second one for the area in which the search for sightseeing will be performed (radius).



Welcome!

Please put your own preferences...
...in other words what we need

In order to find the worh2C places around you, first we need to find YOU! So you must put your Google Public Location Badge ID

Google Public Location Badge ID:	<input type="text"/>
Worth2C area in km:	<input type="text"/>
	<input type="button" value="worth2C"/>

Figure 34. Application's data entry screenshot

7.1.3. PHP

7.1.3.1. About

Since our code is built in PHP language, a brief introduction about this language is necessary.

PHP is recursive acronym for: Hypertext PreProcessor and is a widely-used general-purpose scripting language that is especially suited for Web development, can be embedded into HTML and produce dynamic web pages. What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. [47]



The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. There are three main areas where PHP scripts are used:

- Server-side scripting. This is the most traditional and main target field for PHP. Three things are needed for this: The PHP parser (CGI or server module), a web server and a web browser.
- Command line scripting. A PHP script can be executed without any server or browser. The only thing needed is the PHP parser. This type of usage is ideal for scripts regularly executed using cron (on *nix or Linux) or Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks.
- Writing desktop applications. PHP is probably not the very best language to create a desktop application with a graphical user interface, but if one knows PHP very well, and would like to use some advanced PHP features in one's client-side applications, he can also use PHP-GTK to write such programs.

```
<html>
<head>
  <meta charset="utf-8" />
  <title>PHP Test</title>
</head>
<body>
  <?php
    echo 'Hello World';
  ?>
</body>
</html>
```

7.1.3.2. Syntax

The PHP interpreter only executes PHP code within its delimiters [48]. Anything outside its delimiters is not processed by PHP (although non-PHP text is still subject to

Putting it all together

control structures described within PHP code). The most common delimiters are `<?php` to open and `?>` to close PHP sections. `<script language="php">` and `</script>` delimiters are also available, as are the shortened forms `<?` or `<?='` (which is used to echo back a string or variable) and `?>`. The purpose of all these delimiters is to separate PHP code from non-PHP code, including HTML. The first form of delimiters, `<?php` and `?>`, in XHTML and other XML documents, creates correctly formed XML 'processing instructions'. This means that the resulting mixture of PHP code and other markup in the server-side file is itself well-formed XML.

Variables are prefixed with a dollar symbol, and a type does not need to be specified in advance. Unlike function and class names, variable names are case sensitive. Both double-quoted ("") and `heredoc` strings provide the ability to interpolate a variable's value into the string.

PHP treats newlines as whitespace in the manner of a free-form language (except when inside string quotes), and statements are terminated by a semicolon. PHP has three types of comment syntax: `/* */` marks block and inline comments; `//` as well as `#` are used for one-line comments. The `echo` statement is one of several facilities PHP provides to output text (e.g. to a web browser).

In terms of keywords and language syntax, PHP is similar to most high level languages that follow the C style syntax. If conditions, for and while loops, and function returns are similar in syntax to languages such as C, C++, Java and Perl.

7.1.4. JavaScript

7.1.4.1. About

It is essential to give a brief presentation about JavaScript for there is part of our code that is written in this language which at the moment, is mostly being used to do all kinds of clever (and sometimes annoying) things with pages on the World Wide Web. In other words JavaScript is a scripting language used — in conjunction with HTML — to create interactive Web pages. Contrary to what the name suggests, JavaScript has very little to do with the programming language named Java. The similar name was inspired by marketing considerations, rather than good judgment.

JavaScript has a reputation of being easy to use because the bulk of the document object model (the portion of the language that defines what kind of components, or objects, you can manipulate in JavaScript) is pretty straightforward. For example, if the programmer wants to trigger some kind of event when a person clicks a button, he access the `onClick` event handler associated with the button object.

On the other hand when you load a nice Web page into your browser and wonder how the author created the effect in JavaScript, you can click to view the source code. This source code free-for-all, which is simply impossible with compiled programming languages such as Java, helps you decipher JavaScript programming by example.

Besides being relatively easy, JavaScript is also pretty speedy. Like most scripting languages, it's interpreted (as opposed to being compiled). When one programs using a compiled language, such as C++, he must always reformat, or compile, his code file before he can run it. This interim step can take anywhere from a few seconds to several minutes or more. The beauty of an interpreted language like JavaScript, on the other hand, is that when the programmer makes changes to his code — in this case, to JavaScript script — he can test those changes immediately; he doesn't have to compile the script file first. Skipping the compile step saves a great deal of time during the debugging stage of Web page development.

Another great thing about using an interpreted language like JavaScript is that testing an interpreted script isn't an all-or-nothing proposition, the way it is with a compiled language. For example, if line 10 of a 20-line script contains a syntax error, the first half of the script may still run, and provide feedback immediately. The same error in a compiled program may prevent the program from running at all.

But the greatest of all is that Javascript provides client side scripting. This means that some programmatic behavior is performed inside the client's browser. It does not have to get/post the data back to the server, generate the response there and return it back. [49] [50]

7.1.4.2. Features

Imperative and structured

JavaScript supports much of the structured programming syntax from C (e.g., if statements, while loops, switch statements, etc.). One partial exception is scoping: C-style block-level scoping is not supported. Like C, JavaScript makes a distinction between expressions and statements. One syntactic difference from C is automatic semicolon insertion, in which the semicolons that terminate statements can be omitted. [51]

Dynamic

- *Dynamic typing:* As in most scripting languages, types are associated with values, not with variables. For example, a variable *x* could be bound to a number, then later rebound to a string.

Putting it all together

- *Object based*: JavaScript is almost entirely object-based. JavaScript objects are associative arrays, augmented with prototypes. Object property names are string keys: `obj.x = 10` and `obj['x'] = 10` are equivalent. Properties and their values can be added, changed, or deleted at run-time. Most properties of an object can be enumerated using a `for...in` loop. JavaScript has a small number of built-in objects such as `Function` and `Date`.
- *Run-time evaluation*: JavaScript includes an `eval` function (something of a hybrid between an expression evaluator and a statement executor. It returns the result of the last expression evaluated (all statements are expressions in Javascript), and allows the final semicolon to be left off that can execute statements provided as strings at run-time.

Functional

- *First-class functions*: Functions are first-class; they are objects themselves. As such, they have properties and methods, such as `length` and `call()`; and they can be assigned to variables, passed as arguments, returned by other functions, and manipulated like any other object. Any reference to a function allows it to be invoked using the `()` operator.
- *Nested functions*: "Inner" or "nested" functions are functions defined within another function. They are created each time the outer function is invoked. In addition to that, the scope of the outer function, including any constants, local variables and argument values, become part of the internal state of each inner function object, even after execution of the outer function concludes.
- *Closures*: JavaScript allows nested functions to be created, with the lexical scope in force at their definition, and has a `()` operator to invoke them now or later. This combination of code that can be executed outside the scope in which it is defined, with its own scope to use during that execution, is called a closure in computer science.

Miscellaneous

- *run-time environment* : JavaScript typically relies on a run-time environment (e.g. in a web browser) to provide objects and methods by which scripts can interact with "the outside world". In fact, it relies on the environment to provide the ability to include/import scripts (e.g. HTML `<script>` elements).
- *variadic functions* : An indefinite number of parameters can be passed to a function. The function can access them through formal parameters and also through the local arguments object.

Putting it all together

- *array and object literals* :Like many scripting languages, arrays and objects (associative arrays in other languages) can each be created with a succinct shortcut syntax. In fact, these literals form the basis of the JSON data format.
- *regular expressions*: JavaScript also supports regular expressions in a manner similar to Perl, which provide a concise and powerful syntax for text manipulation that is more sophisticated than the built-in string functions.

Putting it all together

7.1.5. “Show me the Code!”

The part of the full code that retrieves the geographic information from Google Latitude, is the following. [52]

```

1.  $badgeID=$_GET['badgeID'];
2.  $area=$_GET['area'];
3.  //Uses the input user's Badge ID to find the location of the user.
4.  $url="http://www.google.com/latitude/apps/badge/api?user=".$badgeID."&type=json" ;
5.  // We get the content
6.  $content = file_get_contents( $url );
7.  // We convert the user information the JSON
8.  $json = json_decode( $content );
9.  // If the ID is wrong or the Google latitude doesn't have any information
10. if (empty($json->features))
11.  header('location: oops.php');
```

In the first two commands, the variables `$badgeId` and `$area` are transferred from the introduction page to the page that will perform the search. The variables are concatenated with a string that contains the URL address of the Google badge in the variable `$url` in line 4. In line 5 we get the user's geographic information from the URL to the variable `$content` with the php command `file_get_contents`. After that we convert this information into JSON formatted data which is stored in the variable `$json`. If the user's Badge ID is wrong or the Google latitude doesn't have any information, to avoid the program from “crashing” (terminating unexpectedly), an error page is presented (lines 10 and 11).

One may ask “why JSON?”. The scope of our thesis is not to talk about JSON, nevertheless we will discuss in brief here the main reasons for our selection [53] [54]. [55].

1. JSON is easier to read than XML both by programmers and computers
2. JSON requires less tags than XML is lighter weight – XML items must be wrapped in open and close tags whereas JSON the tag is named once
3. Because JSON is transportation-independent the XML `HttpRequest` object for getting the data can be bypassed.
4. As JSON is JavaScript it is easy to put methods and all sorts of goodies in JSON format.
5. One can get JSON data from anywhere, not just his own domain.
6. Easier to combine with AJAX.

Putting it all together

The following example [55] shows the JSON representation of an object that describes a person. The object has string fields for first name and last name, a number field for age, contains an object representing the person's address, and contains a list (an array) of phone number objects.

```
{
  "firstName": "John",
  "lastName" : "Smith",
  "age"      : 25,
  "address"  :
  {
    "streetAddress": "21 2nd Street",
    "city"         : "New York",
    "state"        : "NY",
    "postalCode"   : "10021"
  },
  "phoneNumber":
  [
    {
      "type" : "home",
      "number": "212 555-1234"
    },
    {
      "type" : "fax",
      "number": "646 555-4567"
    }
  ]
}
```

The following XML example carries the same information as the JSON example above.

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber type="home">212 555-1234</phoneNumber>
  <phoneNumber type="fax">646 555-4567</phoneNumber>
</person>
```

Back in our code, the JSON formatted data that is revealed from the Google Latitude API has the following structure:

```
stdClass Object
( [type] => FeatureCollection
[features] => Array ( [0] =>
  stdClass Object ( [type] => Feature [geometry] =>
    stdClass Object ( [type] => Point [coordinates] =>
```

Putting it all together

```

        Array ( [0] => 23.6798892
                [1] => 37.9530796 ) )
[properties] => stdClass Object (
    [id] => -845643805XXXXXX
    [accuracyInMeters] => 30
    [timeStamp] => 1326661412
    [reverseGeocode] => Μοσχάτο, Ελλάδα
    [photoUrl] => http://www.google.com/latitude
    /apps/badge/api?type=photo&photo=swFJ8TQBAAA.0xyyZBdDSupLIrTN1QL
    ABw.bFwdoAWggzP_X6OqdFYRGQ
    [photoWidth] => 96
    [photoHeight] => 96
    [placardUrl] => http://www.google.com/latitude/apps/badge
    /api?type=photo_placard&photo=swFJ8TQBAAA.0xyyZBdDSupLIrTN
    1QLABw.bFwdoAWggzP_X6OqdFYRGQ&moving=false&stale=true&lod=
    1&format=png
    [placardWidth] => 56 [placardHeight] => 59) ) )
stdClass Object
( [type] => FeatureCollection
  [features] => Array ( ) )

```

From the above data we are going to use the longitude, the latitude, the time that the coordinates were recorded and the place that the user is. The part of the full code that does this is given below.

1.	<code>// retrieving the coordinates and other data of the user</code>
2.	<code>\$coord = \$json->features[0]->geometry->coordinates;</code>
3.	<code>\$latitude=\$coord[1];</code>
4.	<code>\$longitude=\$coord[0];</code>
5.	<code>\$timeStamp = \$json->features[0]->properties->timeStamp;</code>
6.	<code>\$whereAreU = \$json->features[0]->properties->reverseGeocode;</code>

In line 1 we store in the array named `$coord` the latitude and the longitude which are in an array in `coordinates` under the `geometry` which belong to `features`. Then, in lines 3 and 4 we separate the array into the variables `$latitude` and `$longitude`. In variables `$timeStamp` and `$whereAreU` (lines 5 and 6) we stored in the same way the `timeStamp` and the `reverseGeocode` which are in the `properties` under the `features`.

7.2. Step 2. – Searching the Semantic Web

Having the location of the user, the next step is to search a corresponding dataset that contains information about the places near the user that are worth seeing such as museums, ancient monuments etc. This dataset is DBpedia [3].

7.2.1. Inside DBpedia – Specific Vocabulary Elements

Semantic web is in many ways unexplored and to a newbie most of the time is incomprehensible. In order to construct an efficient search in DBpedia, we have worked backwards. Acropolis Museum is a museum that by all means is a worth2C place. Therefore we visit DBpedia's page about the museum in http://dbpedia.org/page/Acropolis_Museum to see how the information is represented.

The first thing to notice is the abstract of the museum which value is stored in the `abstract` variable that belongs to the DBpedia ontology. In Figure 35. DBpedia's sample abstract property the value of the abstract is presented in various languages.

Property	Value
<code>dbpedia-owl:abstract</code>	<ul style="list-style-type: none"> Das Akropolismuseum ist ein Museum in Athen, das ausschließlich Fundstücke und Objekte von der Akropolis von Athen präsentiert. Das Museum wurde von Bernard Tschumi und Michalis Fotiadis entworfen und am 20. Juni 2009 offiziell eröffnet. El Museo de la Acrópolis es un museo arqueológico situado en la ciudad de Atenas, a escasa distancia de la Acrópolis. Es uno de los principales museos arqueológicos de Atenas y es también considerado entre los más importantes del mundo, sin embargo, es relativamente pequeño, y por eso el gobierno griego ha decidido trasladarlo fuera de la Acrópolis, dado que anteriormente se situaba dentro de ésta. Fue reinaugurado el 20 de junio de 2009. Akropolis-museo on nykyaikainen museo Ateenassa. Museo keskittyy Akropolis-kukkulaan liittyvän arkeologisen esineistön säilyttämiseen ja kunnostamiseen. Bernard Tschumin suunnittelema museo avattiin virallisesti 20. kesäkuuta 2009 Akropoliin etelärinteellä. Museoesineiden siirto Akropoliin huipulla sijaitsevista vanhasta museosta uuteen alkoi 14. lokakuuta 2007 kestäen kuuden viikon ajan. Uuden museon suunnittelussa lähtökohtana oli, että lordi Elginin 1800-luvun alussa Parthenon-temppelistä ja muualta Akropoliilta viemät, nykyisin British Museumissa säilytettävät marmorneiostokset saataisiin takaisin Kreikkaan. British Museumissa säilytettävien veistosten kopiot ovat esillä Akropolis-museossa, joskin tarvitään alkuperäisveistoksia valkeampina. Näin halutaan kiinnittää musevieraiden huomion siihen, ettei British Museum ole lukuista pyynnöistä huolimatta suostunut palauttamaan marmorneiostoksia alkuperäismaahansa. Il Museo dell'Acropoli (con sede sull'Acropoli di Atene e costruito a fine Ottocento) raccoglie esclusivamente materiali rinvenuti sull'Acropoli: nucleo principale della collezione sono le statue e i frammenti di decorazione architettonica arcaica profanati dai persiani nel 480 a.C., cui si aggiungono sculture del periodo classico. Data la ricchezza delle collezioni, nel 2002 sono cominciati i lavori di costruzione di una nuova e più ampia sede alle pendici dell'Acropoli. Nel giugno 2007 la vecchia sede è stata chiusa per permettere la ricollocazione della collezione nella nuova, più ampia e moderna sede del museo, in corso di completamento nella città bassa, ai piedi dell'acropoli, la cui apertura è stata effettuata a giugno 2009. Il nuovo museo dell'Acropoli al suo interno riunisce - in uno spazio ricostruito delle esatte dimensioni e orientamento del Partenone - tutte le parti del fregio appartenenti al governo greco e quelle in corso di restituzione. Il governo greco e buona parte della comunità internazionale richiedono infatti da molti anni la restituzione dei marmi di Elgin, la parte più consistente delle decorazioni del Partenone all'estero, attualmente in esposizione al British Museum di Londra. The Acropolis Museum is an archaeological museum focused on the findings of the archaeological site of the Acropolis of Athens. The museum was built to house every artifact found on the rock and on its feet, from the Greek Bronze Age to Roman and Byzantine Greece. It also lies on the archaeological site of Makrygianni and the ruins of a part of Roman and early Byzantine Athens. The museum was founded in 2003, while the Organisation of the Museum was established in 2008. It opened to the public on June 21, 2009. Nearly 4,000 objects are exhibited over an area of 14,000 square metres. The Organisation for the Construction of the new museum is chaired by Aristotle University of Thessaloniki Professor Emeritus of Archaeology, Dimitrios Pandermalis. アクロポリス博物館は、アテネのアクロポリスの発掘現場から出土したものを中心に収蔵・展示している考古博物館である。アクロポリスの岩山の上およびその麓で出土した工芸品などを収蔵しており、青銅器時代からローマ属州時代、東ローマ帝国時代のものが中心である。また、Makrygianniの考古遺跡、ローマ時代や東ローマ時代の遺構の上に建てられている。この建物自体は2003年に着工され、2008年に博物館の組織が結成された。開館は2009年6月21日。14,000平方メートルの展示スペースに4千点近くが展示されている。建設プロジェクトを指揮したのはテッサロニキ・アリストテレス大学の考古学名誉教授 Dimitrios Pandermalis で、現館長である。 Het Akropolismuseum is het museum in Athene waar de archeologische vondsten van de Akropolis en zijn hellingen worden tentoongesteld. Nadat het oude Akropolismuseum in juni 2007 werd gesloten, ging het nieuwe Akropolismuseum op 20 juni 2009 open. Dit museum ligt aan de voet van de Akropolis, aan de Dionysiou Areopagitoustraat.

Figure 35. DBpedia's sample abstract property

The label of the entry is also needed in our application software and it can be located under the variable `label` that belongs to the `rdf` schema. Again its value is recorded in many languages.

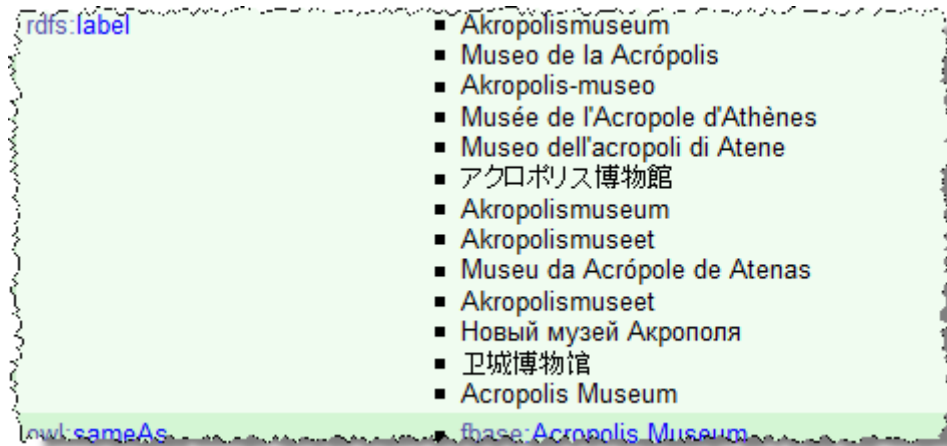


Figure 36. DBpedia's sample label property

The most important to our search is the DBpedia predicate `category` which belongs to the term `subject`. In order to find tourist attraction monuments, the user must define a specific category. For our visualization we choose Greek culture.

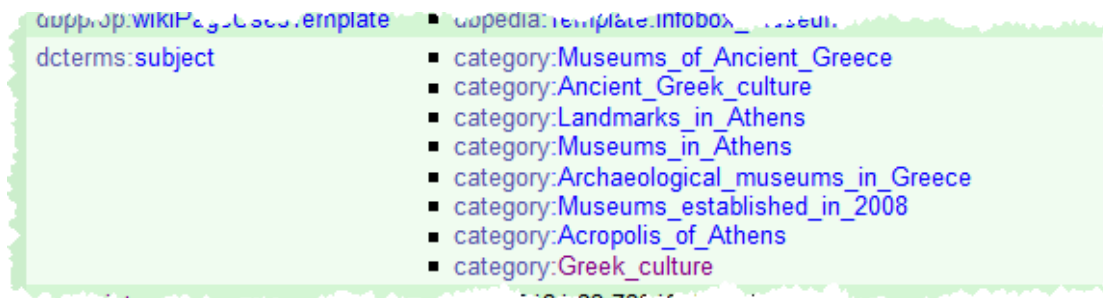


Figure 37. DBpedia's sample subject property

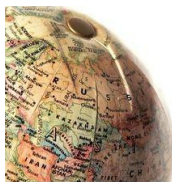
In order to check what this category contains we visit the category's page in the DBpedia in http://dbpedia.org/page/Category:Greek_culture. We can see there that Greek culture belongs to a broader unit of category terms such as Greece, European culture and Culture by nationality.

Moreover we notice that Acropolis of Athens, National Hellenic Museum, Cinema of Greece and many others are subjects of the category Greek culture. More information about categories and sub categories of the resource Greece are found in the APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia.

skos:broader	<ul style="list-style-type: none"> category:Greece category:European_culture category:Culture_by_nationality
skos:prefLabel	Greek culture
is dct:subject of	<ul style="list-style-type: none"> dbpedia:Acropolis_of_Athens dbpedia:White_Tower_of_Thessaloniki dbpedia:Twelve_Olympians dbpedia:Ancient_Greek_clubs dbpedia:Famous_Macedonia dbpedia:Clean_Monday dbpedia:Thessaloniki_International_Film_Festival dbpedia:Petros_(pelican) dbpedia:Rouketopolemos dbpedia:Museum_of_the_Center_for_the_Acropolis_Studies dbpedia:Trata dbpedia:National_Hellenic_Museum dbpedia:Stephen_of_Alexandria dbpedia:Apollon_Theatre_(Patras) dbpedia:Culture_of_Greece dbpedia:Archaeological_Museum_of_Thessaloniki dbpedia:Cinema_of_Greece dbpedia:Greek_festival dbpedia:Heraklion_Archaeological_Museum dbpedia:Artforum_Culture_Foundation
geo:lat	<ul style="list-style-type: none"> 37.967999 (xsd:float) 37.968422 (xsd:float)
geo:long	<ul style="list-style-type: none"> 23.728483 (xsd:float) 23.729000 (xsd:float)

Figure 38. Geo-coordinates of Acropolis Museum

7.2.2. Distance between coordinates



Latitude and longitude are imaginary lines that form a grid on the earth's surface. For centuries, these lines have been indispensable navigational aids for sailors and others who need to plot their exact position on the globe. Because earth is approximately spherical, you can easily calculate the distance between any latitude lines. However, if you are simply planning a trip or a hike, you may find it more helpful to know the number of kilometers (or miles) you will be covering, rather than the exact number of degrees of latitude.

Suppose that we want to find the distance between two latitude lines [56]. First we find the circumference of the earth. Traveling all the way around the world by the longest north-south route, we would cross 180 degrees of latitude on the first half of our trip and 180 degrees on the second half. This journey would cover approximately 40,700 km and 360 degrees of latitude. Then we divide earth's circumference (40,700 km) by 360 to find the distance each latitude degree covers. This works out to roughly 113 km per degree of latitude.

After that we subtract the latitude with the lower numerical value from the latitude with the higher numerical value (for example, if we are traveling from 45 degrees north to 48 degrees north, we subtract 45 from 48). If our journey will take us across

Putting it all together

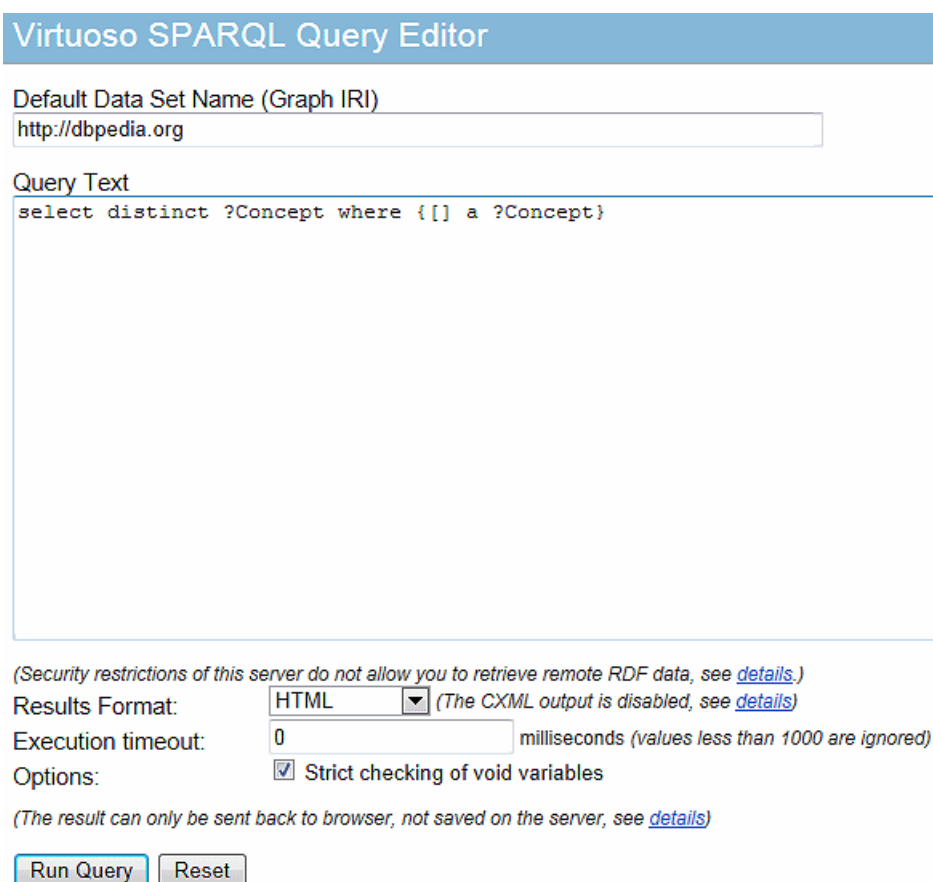
the equator, we use negative numbers to represent latitudes in the southern hemisphere (for example, if we are traveling from 2 degrees south to 1 degree north, we subtract -2 from 1, giving us a total of 3). Last we multiply the difference between our origin and destination latitudes by 113 to calculate the distance between these lines in kilometers. If we prefer to know the distance in miles, we divide the result by 1,6.

In DBpedia latitude and longitude are represented with a 32 bit floating point- the expression `xsd:float`.

7.2.3. “Show me the Code!”

7.2.3.1. Virtuoso – A DBpedia endpoint

In this section we will examine *Virtuoso SPARQL Query Editor*, a DBpedia endpoint that we will use for the purpose of this thesis (see also section 4.7.2 Querying a Public Data Source). Virtuoso can be found at the following URL: <http://dbpedia.org/sparql>.



Virtuoso SPARQL Query Editor

Default Data Set Name (Graph IRI)
http://dbpedia.org

Query Text
select distinct ?Concept where {[] a ?Concept}

(Security restrictions of this server do not allow you to retrieve remote RDF data, see [details](#).)

Results Format: HTML (The CXML output is disabled, see [details](#))

Execution timeout: 0 milliseconds (values less than 1000 are ignored)

Options: ☒ Strict checking of void variables

(The result can only be sent back to browser, not saved on the server, see [details](#))

Run Query Reset

Figure 39. Virtuoso - A DBpedia SPARQL endpoint

Virtuoso has been developed by OpenLink Software with Kingsley Uyi Idehen and Orri Erlingas as the chief software architects. The Framework of Virtuoso is an object-

Putting it all together

relational database SQL. It is available as a commercial product and open source software without support. It provides several pre-built tools that can be tailored to an application that is written in C #, Microsoft. NET, Java, C and C + +.

OpenLink Virtuoso is the first CROSS PLATFORM Universal Server to implement Web, File, and Database server functionality alongside Native XML Storage, and Universal Data Access Middleware, as a single server solution. It includes support for key Internet, Web, and Data Access standards such as: XML, XPATH, XSLT, SOAP, WSDL, UDDI, WebDAV, SMTP, SQL-92, ODBC, JDBC, and OLE-DB. Virtuoso currently supports the following Windows 95/98/NT/2000, Linux (Intel, Alpha, Mips, PPC), Solaris, AIX, HP-UX, Unixware, IRIX, Digital UNIX, DYNIX/PTX, FreeBSD, SCO, MacOS X.

It provides transparent access to existing data sources, which are typically databases from different database vendors. Virtuoso simplifies the process of creating XML data from existing HTML, syndicated XML, and SQL databases. It also enables real-time creation of Dynamic XML documents (DTD or XML Schema based) from homogeneous or heterogeneous SQL Databases "on the fly".

The unique hybrid server architecture of Virtuoso enables it to offer traditionally distinct server functionality within a single product offering that covers the following areas:

- Relational Data Management
- RDF & XML Data Management
- Free Text Content Management & Full Text Indexing
- Document Web Server
- Linked Data Server
- Web Application Server
- Web Services Deployment (SOAP or REST)

Virtuoso has an embedded web server that accommodates Web pages written in either web Virtuoso language (VSP), as well as in PHP, ASP and others. The OpenLink Virtuoso server cares for the performance of RDF data via an associated data interface and a SPARQL endpoint. The RDF Data can be stored directly in Virtuoso or created on the fly from a non-RDF relational database according to a simple match.

Virtuoso Sponger is an RDFizer introduced in Virtuoso 5.0 and provides built-in RDF middleware for transforming non-RDF data into RDF "on the fly". Virtuoso Sponger is a framework for developing wrappers and linked data around different types of data sources. The data sources can range from HTML pages that contain structured data to

Putting it all together

Web APIs. The benefits of Virtuoso Sponger are; the majority of the world's data that resides in a non-RDF form at the current time. Also, the Sponger provides a "Swiss army knife" for RDF structured data generation from non-RDF sources, extracts data from non-RDF Web sources and converts them to RDF. It helps "bootstrap" the Semantic Web, drives the transition of the traditional Document-Web into the emerging Semantic Data-Web and exposes the data in a canonical form for querying and inference. [57]

7.2.3.2. Deep into SPARQL

1.	PREFIX	rdfs:	<http://www.w3.org/2000/01/rdf-schema#>
2.	PREFIX	dbo:	<http://dbpedia.org/ontology/>
3.	PREFIX	dcterms:	<http://purl.org/dc/terms/>
4.	PREFIX	skos:	<http://www.w3.org/2004/02/skos/core#>
5.	PREFIX	geo:	<http://www.w3.org/2003/01/geo/wgs84_pos#>
6.	SELECT DISTINCT		
		?subject	(sql:SAMPLE(?subject) AS ?subjct)
			(sql:SAMPLE(?lat) AS ?lat)
			(sql:SAMPLE(?long) AS ?long)
			(sql:SAMPLE(?label) AS ?label)
			(sql:SAMPLE(?abstract) AS ?abstract)
7.	WHERE	{?subject	geo:lat ?lat;
			geo:long ?long;
			rdfs:label ?label;
			dbo:abstract ?abstract.
		{?subject	dcterms:subject category:Greek_culture}
8.	UNION	{?_category	skos:broader category:Greek_culture.
		?subject	dcterms:subject ?_category.}
9.	UNION	{?_category	skos:broader
			[skos:broader category:Greek_culture].
		?subject	dcterms:subject ?_category.}
10.	FILTER(xsd:float(?lat)-xsd:float(".\$myLat.")	<= \$area2C
	&&	xsd:float(".\$myLat.")-xsd:float(?lat)	<= \$area2C
	&&	xsd:float(?long)-xsd:float(".\$myLong.")	<= \$area2C
	&&	xsd:float(".\$myLong.")-xsd:float(?long)	<= \$area2C
	&&	lang(?abstract)='en'	
	&&	lang(?label)='en').}	
11.	GROUP BY	?subject	ORDER BY ?subject LIMIT 100

Putting it all together

The code presented in the above table extracts the necessary data from the open data stores, where they are stored in RDF format. The first lines of the code with the `prefix` tag denote the datasets that will be used and that contain the resources that will be combined in order to produce the desired results. The datasets are declared through the use of specific URI's for each namespace (see also section 4.6 Some well-known initiatives. The DBpedia dataset contains also geo-coordinates for 697,000 geographic locations. Geo-coordinates are expressed using the W3C Basic Geo Vocabulary. This is a basic RDF vocabulary that provides the Semantic Web community with a namespace for representing lat(itude), long(itude) and other information about spatially-located things, using WGS84 as a reference datum. Use of the geo-coordinates in the code is done through the `geo` prefix. [58] [59]

In the `SELECT` line of code the variables that will hold the results of the query are declared. For the purpose of this thesis five variables are used. Two of them contain the geographical information of the resources (`?lat` and `?long`) while the remaining three (`?subject`, `?label` and `?abstract`) contain the information that will be presented in the visualization process described in the following sections.

At this point, for clarification purposes, the functionality of the `SELECT` command will be further analyzed. The `SELECT` line of code consists of various parts that have the form (`sql: SAMPLE(?variableA) AS ?variableB`). The aim of the command is to store in `variableB` a sample of the values bound to `variableA`. This command is used in conjunction with the command `GROUP BY` that is shown at the last line of the code. The `GROUP BY` command lets the user group sets of data together to perform aggregate functions such as subtotal calculation on each group or, in this thesis' case, select a random value from every group.

We used the `GROUP BY` command as described above because in the testing phase of the application it was evident that there were multiple returns for the same resource due to multiple property values of that resource. Specifically some resources like the Acropolis Museum that was presented in section 7.2.1 Inside DBpedia – Specific Vocabulary Elements, have multiple values for latitude and longitude (see Figure 38. Geo-coordinates of Acropolis Museum). The resulting dataset contains the Cartesian product of the value sets, as shown in the following table which is part of an intermediate query results table.

subject	lat	long
http://dbpedia.org/resource/Acropolis_Museum	37.96842193603516	23.72900009155273
http://dbpedia.org/resource/Acropolis_Museum	37.96842193603516	23.72848320007324
http://dbpedia.org/resource/Acropolis_Museum	37.96799850463867	23.72900009155273
http://dbpedia.org/resource/Acropolis_Museum	37.96799850463867	23.72848320007324

Putting it all together

By using the `(sql:SAMPLE(?variableA) AS ?variableB)` and `GROUP BY` commands all duplicate values have been eliminated.

The `DISTINCT` solution modifier used in the `SELECT` line of code eliminates duplicate solutions. Specifically, each solution that binds the same variables to the same RDF terms as another solution is eliminated from the solution set. The specific modifier could not work efficiently in the previously discussed problem of redundant results because `DISTINCT` only eliminates exactly duplicate answers which was not our case since the combination of the 5 variables (`?subject`, `?label`, `?abstract`, `?lat` and `?long`) was always unique. Nevertheless, the modifier is used just in case!

In the `WHERE` section of code the basic graph pattern to match against the declared datasets is provided. Starting backwards, the lines of code denote that we are searching in DBpedia for those RDF triples that have the object value of `Greek_culture` (or whatever value the user selects from the 2nd level categories that have been identified under resource `Greece` forming the hierarchy presented in APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia). The triples that meet the specific requirements will be combined with the relevant triples that hold the geographical information (Figure 40. Combination of RDF triples).

Putting it all together

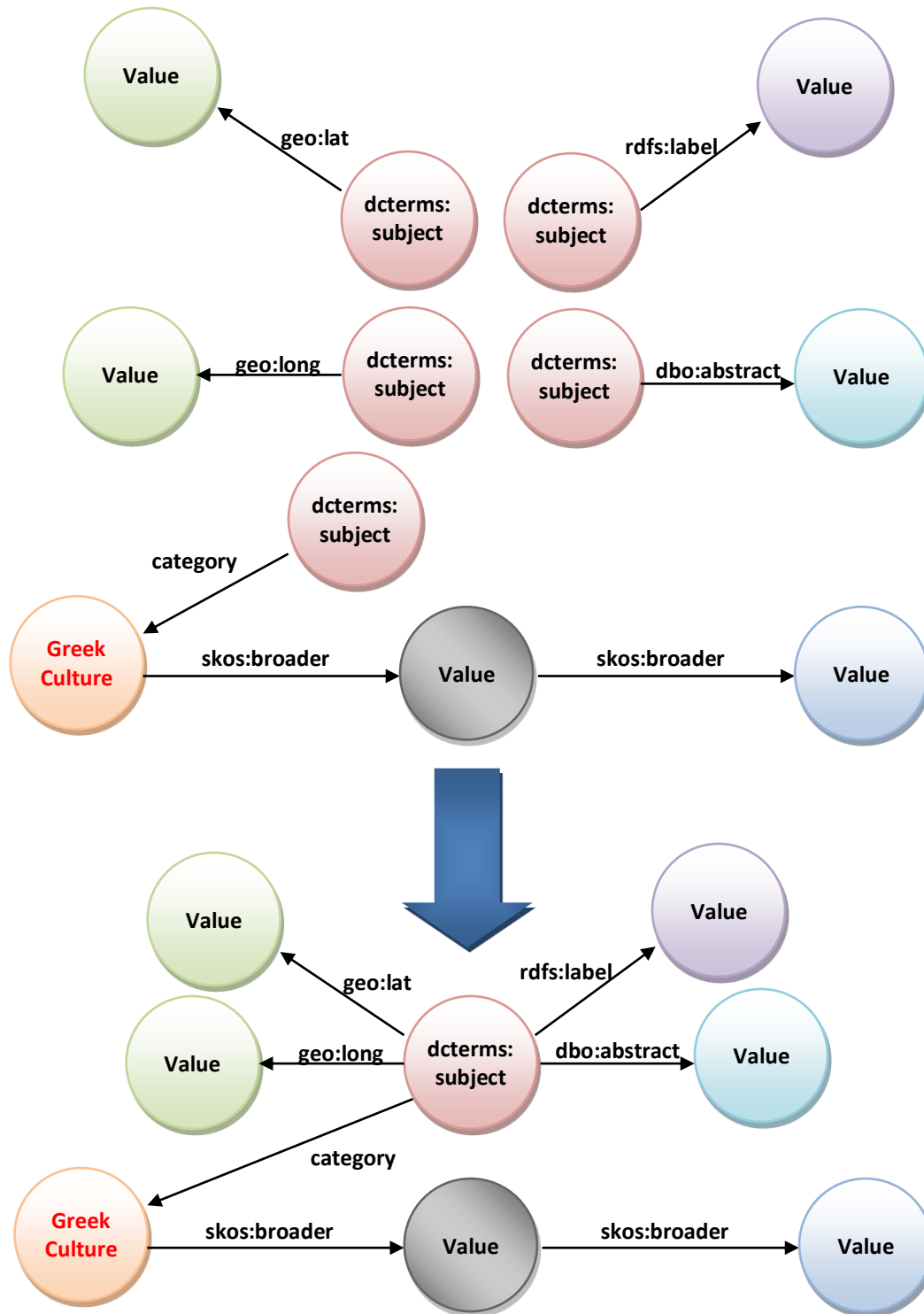


Figure 40. Combination of RDF triples

The UNION statements described in the table at the beginning of this section are a more compact form of the initial code presented below which entails redundancy.

```
WHERE {
  {?subject      dcterm:subject      category:Greek_culture;
    geo:lat      ?lat;
```

Putting it all together

```

        geo:long                ?long;
        rdfs:label              ?label;
        dbo:abstract            ?abstract.
    }
    UNION
    { ?_category      skos:broader      category:Greek_culture .
      ?subject        dcterms:subject  ?_category ;
        geo:lat        ?lat;
        geo:long       ?long;
        rdfs:label     ?label;
        dbo:abstract   ?abstract.
    }
    UNION
    { ?_category      skos:broader      [skos:broader
category:Greek_culture] .
      ?subject        dcterms:subject  ?_category .
        geo:lat        ?lat;
        geo:long       ?long;
        rdfs:label     ?label;
        dbo:abstract   ?abstract.
    }
}

```

The semicolon (;) at the end of most lines has the meaning that we described back in the N3 section (4.3.3.2 Notation 3 (N3) syntax) which is “here comes another predicate and object to go with this triple’s subject.”

The results that are returned with the procedure described in the previous paragraph refer to triples that have the object value of `Greek_culture`. As it was mentioned in section 4.3.3 Triple view of RDF statements this object by itself can act as subject (resource) to other triples forming the hierarchy that is presented in APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia. But the query described so far does not return the resources contained in the subcategories of the `Greek_culture` hierarchy.

For this reason the `UNION` statements of lines 10-15 of the code are used. The idea is to use a procedure, similar to the one described above, to retrieve suitable triples that belong to lower levels of the hierarchy beginning from `Greek_culture`. This is accomplished with the use of the term `skos:broader` of the resource `Greek_culture`, returning also results from two levels underneath (subcategories) the starting point. The variable `?_category` is an internal one used only as a

Putting it all together

temporary placeholder of the returned results by the use of the `skos:broader` term. This is the reason that it is not declared in the `SELECT DISTINCT` line.

In the `FILTER` part of the code (lines 16-21) restrictions are imposed to the values of the returned data. The first restriction has to do with the distance of the resources from the user's position. The distance is calculated by subtracting the values of the resources' latitude and longitude (`?lat` and `?long` variables) from the corresponding values of the user's position (`$myLat` and `$myLong` variables) following the procedure that was described in section Distance between coordinates. Results that are in greater distance than the one defined by the user and stored in the `$area2C` variable are discarded.

The other type of restrictions deal with the language of the returned resources. Specifically, only triples with `abstract` and `label` values in English are permitted.

Finally, the remaining results are sorted using the `ORDER BY` clause which establishes the order of the returned results according to the contents of the `?subject` variable. The final part of the command `LIMIT` limits the number of the returned results to the integer specified.

The following table shows the first two rows of the results that the SPARQL query returns when it is executed.

Putting it all together



subject	lat	long	label	abstract
http://dbpedia.org/resource/Acropolis_Museum	37.96842193603516	23.72900009155273	"Acropolis Museum"@en	"The Acropolis Museum is an archaeological museum focused on the findings of the archaeological site of the Acropolis of Athens. The museum was built to house every artifact found on the rock and on its feet, from the Greek Bronze Age to Roman and Byzantine Greece. It also lies on the archaeological site of Makrygianni and the ruins of a part of Roman and early Byzantine Athens. The museum was founded in 2003, while the Organisation of the Museum was established in 2008. It opened to the public on June 21, 2009. Nearly 4,000 objects are exhibited over an area of 14,000 square metres. The Organisation for the Construction of the new museum is chaired by Aristotle University of Thessaloniki Professor Emeritus of Archaeology, Dimitrios Pandermalis."@en
http://dbpedia.org/resource/Acropolis_of_Athens	37.97142028808594	23.72616577148438	"Acropolis of Athens"@en	"The Acropolis of Athens or Citadel of Athens is the best known acropolis (Gr. akros, akron, edge, extremity + polis, city, pl. acropoleis) in the world. Although there are many other acropoleis in Greece, the significance of the Acropolis of Athens is such that it is commonly known as The Acropolis without qualification. The Acropolis was formally proclaimed as the preeminent monument on the European Cultural Heritage list of monuments on 26 March 2007. The Acropolis is a flat-topped rock that rises 150 m (490 ft) above sea level in the city of Athens, with a surface area of about 3 hectares. It was also known as Cecropia, after the legendary serpent-man, Cecrops, the first Athenian king."@en

7.2.3.3. Transform SPARQL results into JSON

All the SPARQL's Query is stored in our code in a variable named `$query`. This variable is concatenated in the variable `$searchUrl` along with the SPARQL's endpoint in DBpedia and the JSON format (which is saved in the variable `$format`). The PHP command that does this string concatenation is the following:

```
$searchUrl = 'http://dbpedia.org/sparql?'
            . 'query='.urlencode($query)
            . '&format='.$format;
```

Next thing to do is to send this URL over HTTP using PHP. In order to send a HTTP request for a URL in PHP we are using a library named `cURL`. `cURL` allows the application to connect and communicate with many different types of servers and with various types of protocols. [60]. The part of the code for the HTTP request is as follows:

```
function request($url) {
    // is curl installed?
    if (!function_exists('curl_init')){
        die('CURL is not installed!'); }
    // get curl handle
    $ch= curl_init();
    // set request url
    curl_setopt($ch, CURLOPT_URL, $url);
    // return response, don't print/echo
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $response = curl_exec($ch);
    curl_close($ch);
    return $response;}
```

Above is the function `request` that will request our variable `$url` (which will have the same content with the mentioned variable `$searchUrl`) and return the response body as a string. The methods that are used are:

- `curl_close` — Close a cURL session
- `curl_init` — Initialize a cURL session
- `curl_setopt` — Set an option for a cURL transfer (for further information about the parameters in this method, follow the following URL: <http://www.php.net/manual/en/function.curl-setopt.php>)

The request to our URL above will return a string of JSON data because that's the format we specified in the variable `$format`. By using the method `json_decode`. The `json_decode` has a boolean parameter `assoc` which if set to `true` returns a PHP array. With this PHP array we can easily access all the data returned in our response.

```
$responseArray = json_decode(  
    request($requestURL, true);
```

Finally the results of the SPARQL query are transformed with JSON as following:

```
{ "head": { "link": [], "vars": [ "subject", "lat", "long", "label", "abstract" ] },  
  "results": { "distinct": false, "ordered": true, "bindings": [  
    { "subject": { "type": "uri",  
                  "value": "http://dbpedia.org/resource/Acropolis_of_Athens" },  
      "lat": { "type": "typed-literal",  
              "datatype": "http://www.w3.org/2001/XMLSchema#float",  
              "value": "37.97142028808594" },  
      "long": { "type": "typed-literal",  
              "datatype": "http://www.w3.org/2001/XMLSchema#float",  
              "value": "23.72616577148438" },  
      "label": { "type": "literal", "xml:lang": "en",  
                "value": "Acropolis of Athens" },  
      "abstract": { "type": "literal", "xml:lang": "en",  
                   "value": "The Acropolis of Athens or ....." } } ] }
```

7.3. Step 3. – Visualization of the results on the Map

After finding the position of the user and searched the DBpedia for the worth visiting places nearby the user, it is time to visualize all this information to a map. Great resource for that representation is Google Maps API, which is analyzed further on.

7.3.1. Google Maps API V3

As we have mentioned Google Maps provides a flexible way to integrate maps to provide directions, location information provided by the Google Maps Application Programming Interface in a web application. [61] [62] [63] [64]

7.3.1.1. *Connect with the Google Maps API*

The very first and the most important thing to do, is to put the right commands in the code in order to connect with the Google Maps API. The version of the API used is Google Maps API V3 and there are a numbers of reasons why we do so: [64]

- In the earlier versions of the Google Maps API, a developer had to register the web application with Google and get an API key. In version 3, key registration has been eliminated.
- In previous versions of the API there were lots of global variables, distinguishing themselves by having a capital G in the beginning, like `GMap` and `GLatLng`. In the new API the namespace `google.maps` is used. So what was formerly called `GLatLng` is now called `google.maps.LatLng`. Using namespaces is good coding praxis in order to avoid collisions with other code.
- The API V3 makes heavy use of object literals to pass arguments to its methods. The old API used object literals as well, but in this version they've taken it a step further which I think is positive. I think this is a really beautiful way of coding. It's easy to use, easy to maintain and easy to expand.
- To initialize the map in V3 we need to call the constructor of the `Map` class. It takes two arguments the first one being a reference to a HTML element where you want the map to be inserted (often `<div id="map">`) and the second one being an object literal (options) containing a set of properties. The big difference is that in the old API you had to call the method `setCenter()` immediately after calling the constructor in order for the map to `display.setCenter()` told the map what location would be the center of the map and what the initial zoom level would be. In the API V3 this is not necessary since both these properties are passed in the option parameter. So instead of making two calls, you now have to do only one.

Back to our code, the following part is used to connect to the Google Maps API V3:

```
<script  
type="text/javascript"  
src="http://maps.googleapis.com/maps/api/js?sensor=false">  
</script>
```

Some explanation should be given about the variable `js?sensor = false`. This variable commands Google to *not* try to load one's location. This is only useful on mobile devices or geolocation enabled browsers. But we are using the Google Latitude Badge for that so we turn the `sensor` variable to false.

7.3.1.2. Create the Google Map

In order to create our Google Map centered around the user's position we use the following JavaScript code:

```
function initialize(lat,lng) {  
var myLatLng = new google.maps.LatLng(lat,lng);  
var myOptions = {  
    zoom: 10,  
    center: myLatLng,  
    mapTypeId: google.maps.MapTypeId.ROADMAP};  
var map = new  
google.maps.Map(document.getElementById('map_canvas'),  
myOptions);}
```

Map Class

In the above code, we used the Map class which takes options as parameters and refers to the user's latitude and longitude that we had already revealed in step 1. In the following table we can see all the options of the Map class:

Property	Class								
MapTypeControl:true/false	mapTypeControlOptions <table> <tr> <td>Property</td><td>Constants/Values</td></tr> <tr> <td>style</td><td>DEFAULT
HORIZONTAL_BAR
DROPDOWN_MENU</td></tr> <tr> <td>position</td><td>BOTTOM
BOTTOM_LEFT
BOTTOM_RIGHT
LEFT
RIGHT
TOP
TOP_LEFT
TOP_RIGHT</td></tr> <tr> <td>mapTypeIds</td><td>ROADMAP
SATELLITE
Hybrid
Terrain</td></tr> </table>	Property	Constants/Values	style	DEFAULT HORIZONTAL_BAR DROPDOWN_MENU	position	BOTTOM BOTTOM_LEFT BOTTOM_RIGHT LEFT RIGHT TOP TOP_LEFT TOP_RIGHT	mapTypeIds	ROADMAP SATELLITE Hybrid Terrain
Property	Constants/Values								
style	DEFAULT HORIZONTAL_BAR DROPDOWN_MENU								
position	BOTTOM BOTTOM_LEFT BOTTOM_RIGHT LEFT RIGHT TOP TOP_LEFT TOP_RIGHT								
mapTypeIds	ROADMAP SATELLITE Hybrid Terrain								
navigationControl:true/false	navigationControlOptions <table> <tr> <td>Property</td><td>Constants/Values</td></tr> <tr> <td>Position</td><td>BOTTOM
BOTTOM_LEFT
BOTTOM_RIGHT
LEFT
RIGHT
TOP
TOP_LEFT
TOP_RIGHT T</td></tr> <tr> <td>style</td><td>DEFAULT
SMALL
ANDROID</td></tr> </table>	Property	Constants/Values	Position	BOTTOM BOTTOM_LEFT BOTTOM_RIGHT LEFT RIGHT TOP TOP_LEFT TOP_RIGHT T	style	DEFAULT SMALL ANDROID		
Property	Constants/Values								
Position	BOTTOM BOTTOM_LEFT BOTTOM_RIGHT LEFT RIGHT TOP TOP_LEFT TOP_RIGHT T								
style	DEFAULT SMALL ANDROID								
scaleControl:true/false	scaleControlOptions: scalecontroloptions has the same properties as navigation control options (position, style) and behavior is also the same.								
disableDoubleClickZoom: true/false									
scrollwheel: true/false									
draggable: true/false									
streetViewControl: true/false									

For further information about Google Maps API visit the pages of Google code at the following URL:

<http://code.google.com/intl/el/apis/maps/documentation/javascript/reference.html>

Map Marker

In order to point out certain locations in the Map we use markers. In our application there is a marker that highlights the position of the user and others that show the worth2C places. The `Marker` class provides the option to display a marker. The piece of the code that initializes the user's marker is the following:

```
var markerA = new google.maps.Marker({  
    position: myLatLng  
    title: "You."});
```

In the markers we can use the following attributes:

position: it is a mandatory attribute that contains the longitude and the latitude of the place where the marker should appear.

icon: if we don't want to use Google's marker, we should provide the path of the image file for a custom marker

shadow: if we want our marker to have some shadow, here is the path of the image file for the marker's shadow

draggable: with value true or false, we define if we want our marker to move inside the map by using the left click of the mouse

title: we put a text that will appear when someone put the mouse over the marker. In our code the text is "You. "

visible: with value true or false we define if we want our marker to show up in our map.

Info Window

When the marker is displayed on the map, we can create an `onclick` event which provides the user with a popup window showing some information about the place. The piece of the code that creates the user's marker info box is given below:

```
var infowindowA = new google.maps.InfoWindow({  
    content: "You are...HERE!" });  
google.maps.event.addListener(markerA, 'click',  
function() { infowindowA.open(map,markerA);});
```

Multiple Markers

As we have described, the purpose of this thesis is to mark all the interesting spots around the user. In order to manage that, we must use multiple markers on the Google Map.

First thing is to create an array that will contain the coordinates of the marker, the name of the spot and a little piece of information that will appear in an info box. And now let's reveal the big secret: all this data will come from the DBpedia via the SPARQL query that had already executed in step2. To sum up, we already have latitude, longitude, label and abstract of the places that are near by the user. All this are stored in the array `$responseArray` with the following command:

```
$responseArray = json_decode(
    request($requestURL), true);
```

All the information we need is now in the array and with a `for` loop we are going to extract them from the array and store them in a variable. Note that the code is Javascript while the array that holds the data is a PHP variable. In this case we need to embed PHP values inside Javascript code. The part of the code that does exactly that is given below.

```
var locations = [
    <?php for($i=0; $i<10; $i++) { ?>
        ['<?php echo @$responseArray ["results"]
        ["bindings"][$i]["abstract"]["value"];?>',
        <?php echo @$responseArray ["results"]
        ["bindings"][$i]["lat"]["value"];?>,
        <?php echo @$responseArray ["results"]
        ["bindings"][$i]["long"]["value"];?>,
        '<?php echo @$responseArray ["results"]
        ["bindings"][$i]["label"]["value"]; ?>'],
    <?php }?> ]
```

With the `for` loop we managed to create a array in Javascript named `locations` that have 10 lines and 4 columns. Each line refers to one monument we want to spot near the user while the columns contains the abstract, the latitude, the longitude and the label for each of that. Finally we must visualize all the markers in our map.

```
var infowindowB = new google.maps.InfoWindow();
var i;
for (i = 0; i < locations.length; i++) {
    markerB = new google.maps.Marker({
        position: new google.maps.LatLng(locations[i][1],
        locations[i][2]),
        map: map ,
        title:locations[i][3] });
    google.maps.event.addListener(markerB, 'click',
        (function(markerB, i) {
            return function()
            {infowindowB.setContent(locations[i][0]);
            infowindowB.open(map,markerB);} })
        (markerB, i)); }
```


Synopsis - Conclusions

With the above `for` loop we create 10 markers each of one locates in the coordinates given by `locations[i][1]` and `locations[i][2]` while the title of the marker is given by `locations[i][3]`.

Furthermore in the `locations[i][0]` we have stored the abstract which will be appear in an infobox when the user clips the counterpart marker.

All the parts of the code presented so far along with the rest of the code is presented in APPENDIX B – The code.

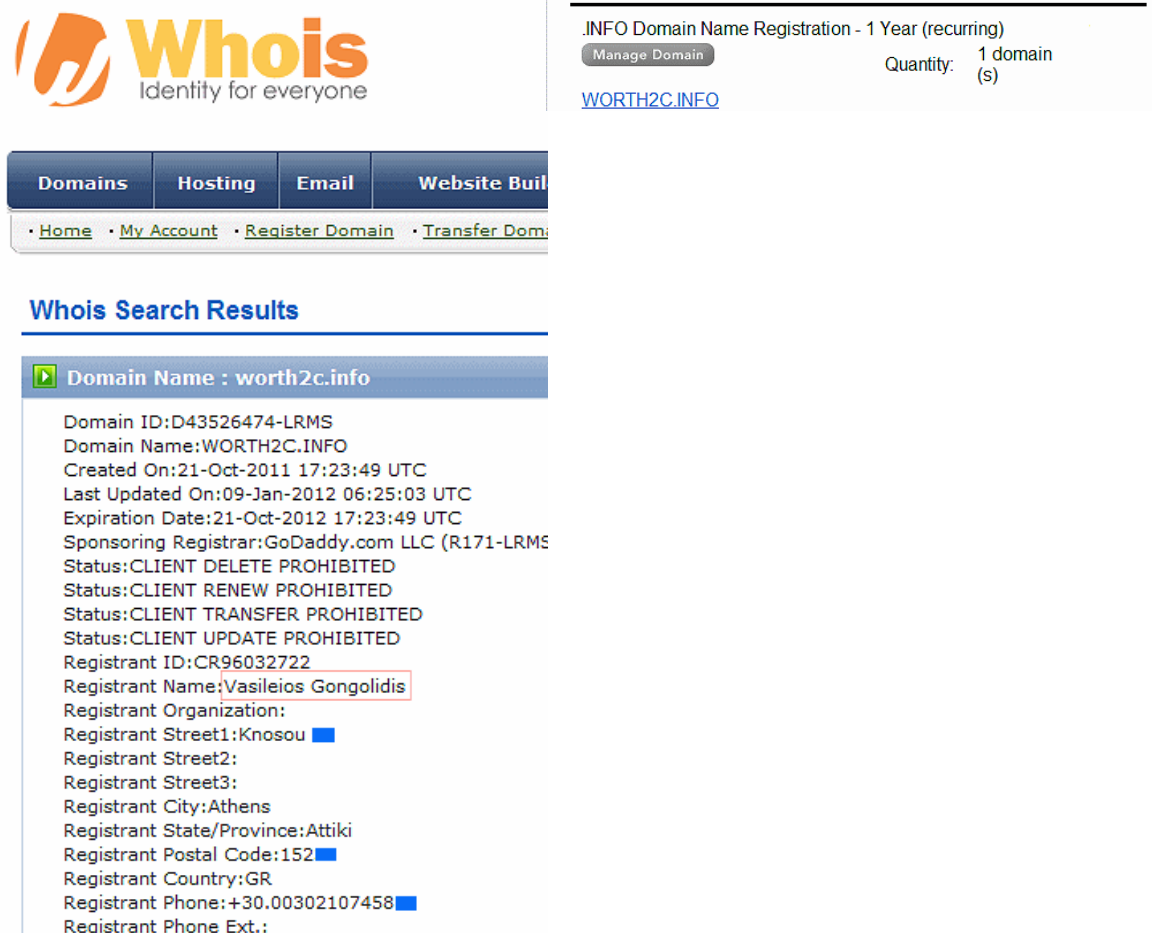
8. Putting it all together

In the following sections of this chapter all the technologies presented so far will be combined and visualized in an application that will be hosted on the website that was created for the specific purpose at the following URL: <http://worth2c.info>.

8.1. Domain Name

For the selection of an appropriate domain name for the website a number of different options were examined. In the end the name WORTH2C was chosen and the registration of the name was made with the GoDaddy provider of the US. [65]

Figure 41. Worth2c.info domain name registration



The figure consists of two main parts. The top part is a screenshot of a GoDaddy email confirmation for a domain purchase. It includes the GoDaddy logo, a 'Thank you for your order!' message, and order details: Customer Number 43824, Username 43824, and Receipt Number 366176017. It also shows the registrant's name, Vasileios Gongolidis, and a link to manage the domain. The bottom part is a screenshot of a Whois search results page for the domain worth2c.info. It lists various domain details including ID, name, creation and expiration dates, registrar information, and registrant details such as name, organization, and contact information.

GoDaddy.com
Thank you for your order!

Customer Number: 43824 Username: 43824 Receipt Number: 366176017

Dear Vasileios Gongolidis,

This email contains important information regarding your recent GoDaddy.com purchase – please save it for reference.

.INFO Domain Name Registration - 1 Year (recurring)
Manage Domain Quantity: 1 domain(s)
WORTH2C.INFO

Whois
Identity for everyone

Domains Hosting Email Website Build

• Home • My Account • Register Domain • Transfer Domains

Whois Search Results

Domain Name : worth2c.info

Domain ID:D43526474-LRMS
Domain Name:WORTH2C.INFO
Created On:21-Oct-2011 17:23:49 UTC
Last Updated On:09-Jan-2012 06:25:03 UTC
Expiration Date:21-Oct-2012 17:23:49 UTC
Sponsoring Registrar:GoDaddy.com LLC (R171-LRMS)
Status:CLIENT DELETE PROHIBITED
Status:CLIENT RENEW PROHIBITED
Status:CLIENT TRANSFER PROHIBITED
Status:CLIENT UPDATE PROHIBITED
Registrant ID:CR96032722
Registrant Name:Vasileios Gongolidis
Registrant Organization:
Registrant Street1:Knosou
Registrant Street2:
Registrant Street3:
Registrant City:Athens
Registrant State/Province:Attiki
Registrant Postal Code:152
Registrant Country:GR
Registrant Phone:+30.00302107458
Registrant Phone Ext.:

Figure 42. Domain worth2c.info WHOIS registration details

8.2. Web hosting plan

The next step was to purchase a hosting plan suitable for the purpose of our application. The requirements were for the hosting plan to support PHP5, have a

APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia

secure Java FTP client for uploading files to the website, DNS control management, user friendly control panel and quick customer support service.



Figure 43. Purchase of web hosting plan

For all the above reasons as well as previous experience, the *GoDaddy* provider was selected.

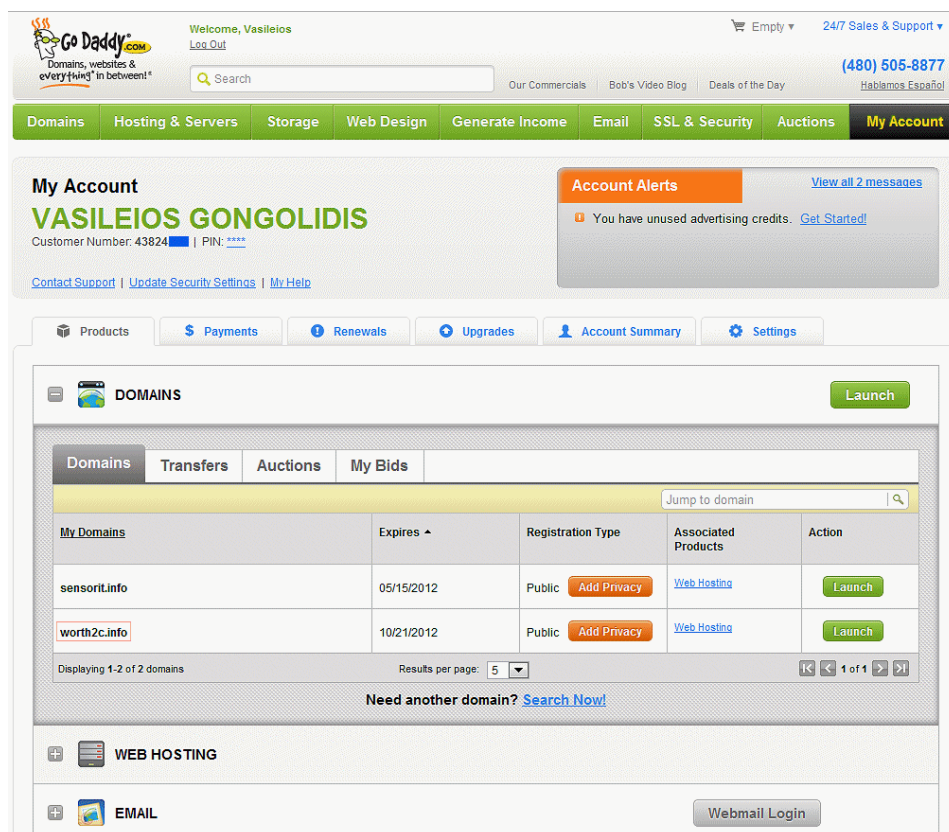


Figure 44. Provider's control panel

8.3. Visualization

The stage has been set. Using the control panel of the hosting plan we uploaded the necessary php, Javascript and css style files. The only thing that remains is to access the application at the following address:

<http://www.worth2c.info>

After entering the required information (Google Latitude Budge and radius of interest) the show begins!

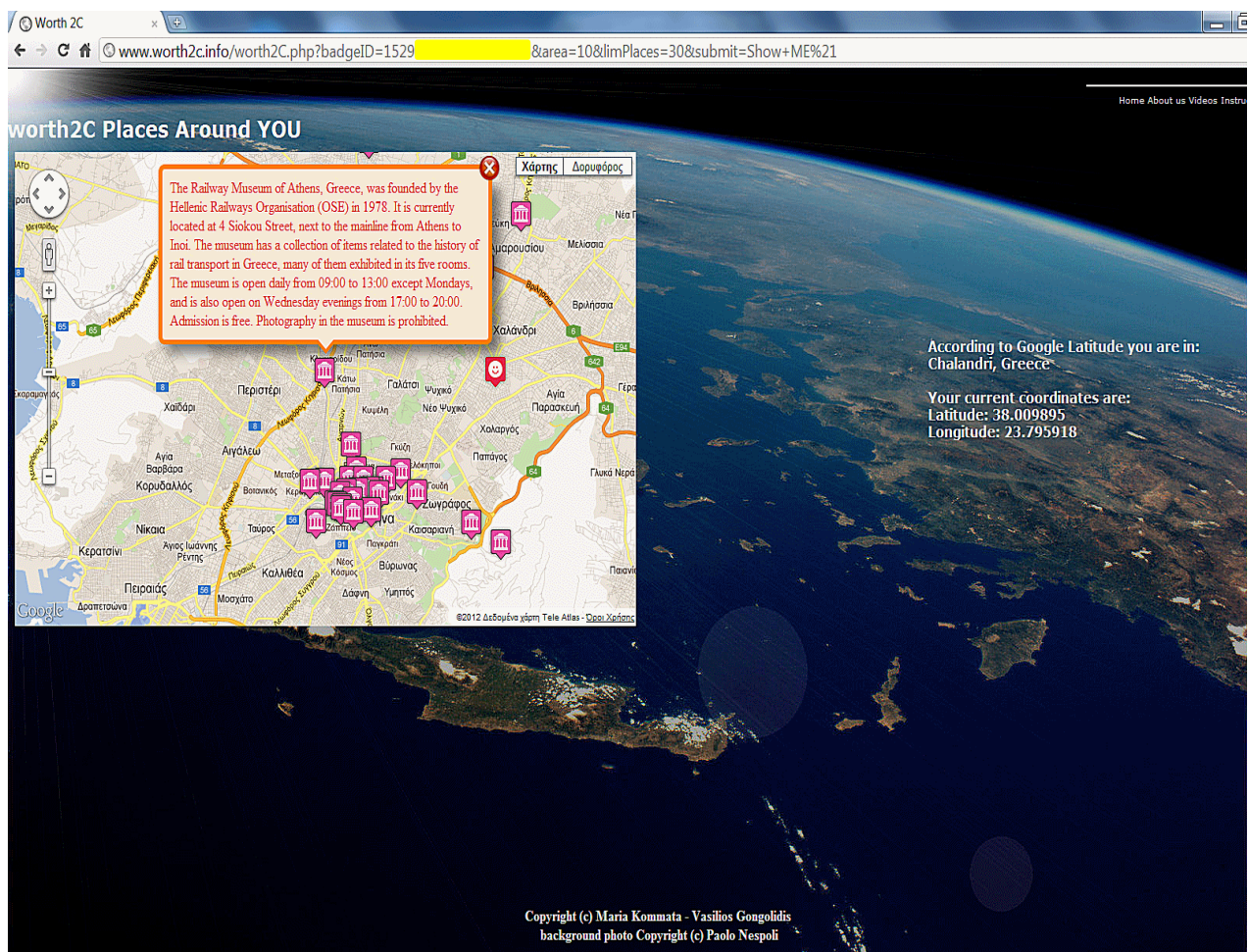


Figure 45. worth2c.info sample page

9. Synopsis - Conclusions

In this thesis we have combined various technologies to build an application that can query the semantic web for information that correspond to a user's criteria. This information is only presented to the user if it is in proximity to his location. The technologies used are:

- **Google Latitude.** Through a technology known as XPS, Google Latitude combines Cellular positioning, Wi-Fi Positioning, and GPS to establish automated location detection. Via their own Google Account, the user's location is mapped on Google Maps.
- **Google Maps.** Google Maps is a web mapping service application and technology provided by Google, that powers many map-based services, including the Google Maps website, and maps embedded on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, car, or public transport and an urban business locator for numerous countries around the world.
- **Geocoding.** Geocoding is the process of converting addresses such as street addresses, or zip codes (postal codes), into geographic coordinates (often expressed as latitude and longitude), which you can use to place markers or position a map. Reverse Geocoding is the process of finding an address, toponym or other type of resource for a given latitude/longitude pair.
- **DBpedia.** DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web.
- **Dublin Core Metadata Initiative (DCMI).** A dataset used to describe a full range of web resources: video, images, web pages etc. and physical resources such as books and objects like artworks. Dublin Core can be used for multiple purposes, from simple resource description, to combining metadata vocabularies of different metadata standards, to providing interoperability for metadata vocabularies in the linked data cloud and semantic web implementations
- **Simple Knowledge Organization System (SKOS).** A dataset designed for representation of thesauri, classification schemes, taxonomies, subject-heading systems, or any other type of structured controlled vocabulary.
- **SPARQL.** SPARQL is a query language for pattern matching against RDF graphs. The syntax resembles SQL, making for a friendly learning curve, but SPARQL is far more powerful, enabling queries spanning multiple disparate (local or remote) data sources containing heterogeneous semistructured data.

- **JSON.** JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. The main reason for choosing JSON over XML is speed. JSON is simple and can represent the exact same data with fewer characters than XML because it uses no tags.
- **PHP.** PHP is a server-side scripting language for creating dynamic Web pages. PHP is Open Source and cross-platform. PHP is especially lightweight and speedy. Without any process creation overhead, it can return results quickly. It offers excellent connectivity to many databases (and ODBC), and integration with various external libraries.
- **Javascript.** JavaScript is an object-oriented scripting language that lets you build interactions between page content, the state of the browser, and the actions of the reader.

The blending of the above mentioned technologies for the purpose of our application was as follows:

1. Using Google Latitude’s Badge that is related to a user’s Google account, the user’s location is identified with precision according to the specification set by the user.
2. All the data that is related to the user’s Google Latitude Badge is returned to the application using the JSON format. This data contains among others the user’s position expressed by a certain latitude and longitude.
3. Another piece of information contained in the returned JSON and revealed by applying reverse Geocoding, is the user’s actual place (e.g. city).
4. The returned latitude and longitude values are fed to a SPARQL query that is executed against a DBpedia endpoint called Virtuoso. The dataset of DBpedia is queried for resources that fall into a certain category indicated by the user and are in a proximity distance also set by the user. By this way, available geographic data is combined with RDF triples stored in DBpedia (therefore with available data in Wikipedia) along with other available vocabularies, like DCMI and SKOS, to formulate unique per user results.
5. These returned results are depicted on a map using Google Maps and specifically Google Maps API V.3. Certain values of the returned results, like the `abstract` value, is presented in infoboxes.
6. The synthesis of the various parts of the application was made using PHP and Javascript as conjunctive material.

The outcome of the combination of the above mentioned technologies can be accessed at <http://www.worth2c.info>.

Let us briefly analyze what has been achieved with this application through a series of questions and answers. We built an application that has the capability of retrieving data from the Semantic Web according to a user’s location and preferences.

- ❓ What is the prerequisite for the application to run and return results? The answer is the existence of data.
- ❓ Is the existence of data a sufficient condition for a third party application to use it, process it and return meaningful results? The answer is no, because the data has to be publicly available or in other words it must be Open Data.
- ❓ Is the existence of Open Data the only prerequisite for the Semantic Web to fulfill its promise? Definitely no, because all available datasets containing Open Data have to be interlinked for humans and machines to pose complex queries and get answers composed of information from various datasets. The linking of Open Data forms the Linked Open Data cloud.
- ❓ How does Linked Open Data deal with the ambiguity that is inherent in the meaning of many terms used by humans and also implemented in computer information systems? By forming vocabularies and ontologies that represent a shared and common understanding of a domain that can be communicated between people and application systems. This is accomplished through the use of RDF and OWL technologies.
- ❓ How can someone retrieve information from this Linked Open Data cloud? The answer is with the use of SPARQL, a language designed to query not relational data, but data conforming to the RDF data model and stored in the “cloud”-the Linked Open Data cloud.

The answers to all the above questions, essentially the answer to the question “*Why Semantic Web?*” is demonstrated with the application developed in the context of this thesis. Information contained in Wikipedia was combined with geographic information to produce a unique user experience. This is only a sample of the benefits gained when querying the Semantic Web. Based on the work of this thesis one can follow many interesting paths.

Since the road has been laid and the technological platform has been developed for addressing the Semantic Web, the only thing that remains to be used is imagination. So, let’s try using some of it!

How about linking our existing application with the Flickr dataset? Flickr is almost certainly the best online photo management and sharing application in the world. It is home to over five billion of the world’s photos uploaded by the users. With our

application in its existing form, a user can be informed of worth to see places that surround him, but does not have any visual information whatsoever of these places. It would be a rather interesting experience if the user could see how these places look like, not according to a guide's, maybe biased, view but according to other visitors' opinion.

How about linking our application with the Friend Of A Friend (FOAF) project? By this way, we can see information about certain individuals depicted on our map. One may argue that we don't have to use the Semantic Web to do this. We can stick to Google Latitude to locate these individuals. That is true but that is also the only thing you can do with Google Latitude. On the other hand, using the FOAF vocabulary provides the opportunity to retrieve various information concerning specific individuals like age, primary topic, e-mail address, Skype ID, Facebook ID and other social media information. You can also learn about other people a specific individual knows.

Now, let's take another turn and try something different. Since my work is on research policy that has a regional aspect involved, I'm interested in available data with geographic characteristics. For example, suppose that I am visiting a city to locate all the innovation drivers from the private sector. These may be enterprises that have benefited from public granting schemes for innovative initiatives they have undertaken. If all the previous information has been described through an appropriate ontology, it will be very easy to locate these companies with a little “sparkling”, without any prior preparation from the office.

The same would be true and interesting with other sectors of the economy like health and generally with all government data. Initiatives like the “*Linking Open Government Data*” project [68], open new horizons in the exploitation of the Semantic Web.

Back to our application, there are a lot of things that could be improved from a technological point of view not related to the Semantic Web. Security issues related to the exposure of a user's personal Google Latitude Badge, when typing this in our site at <http://www.worth2c.info>, could be resolved. Usability would also be increased if the user didn't have to memorize or write down a 19-digit number that corresponds to his Google Latitude Badge, but instead log on to our application and connect to his Google Latitude account which is easier. Another useful feature would be to broaden the categories under which a user can query for information. At the moment, our application is searching for RDF triples that fall under the category *Greek_culture*, but it is very easy and only a matter of time to expand to different categories like, for example *Environment_of_Greece* or *Health_in_Greece*.

APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia



There is no limit to extending the usage of this application but only from lack of imagination. Fortunately, any avocation with the Semantic Web improves imagination!

10.APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia

The structure of the entity (resource) *Greece* within the dataspace *dbpedia.org* is presented in this appendix. Considering Greece to be the first level, we have analyzed the structure of the entity and its “subentities” until 3rd level down. Because of space restrictions only the 3rd level for resource *Greek_culture* is shown in Figure 46. 3-level structure of resource *Greece*. The full analysis for every entity is presented in the following tables.

APPENDIX A – Analysis of resource “Greece” from dataspace DBpedia

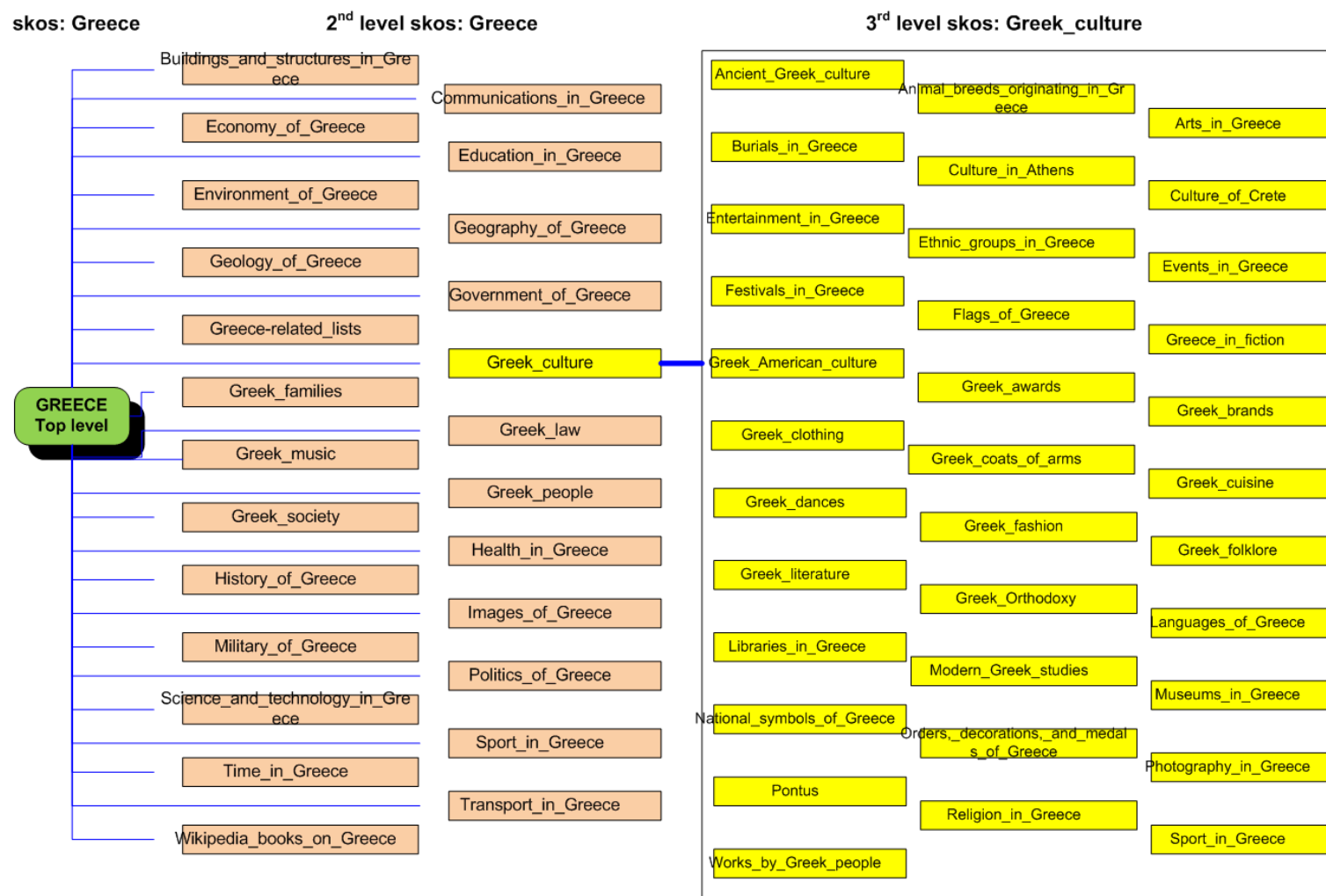


Figure 46. 3-level structure of resource Greece

About: [Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Greece Category:
skos:broader	category:Member states of La Francophonie category:Balkans category:Member states of the European Union category:European countries category:Bicontinental countries category:Member states of the Council of Europe category:Countries of the Mediterranean Sea category:Categories named after countries
skos:prefLabel	Greece
is dcterms:subject of	dbpedia:Doric Greek dbpedia:Famous Macedonia dbpedia:Greece dbpedia:Greek War of Independence dbpedia:Portal:Greece dbpedia:Ohi Day dbpedia:Greeks in Albania dbpedia:Outline of Greece dbpedia:NorthWind II
is skos:broader of	category:Greek music category:Health in Greece category:Environment of Greece category:Images of Greece category:Transport in Greece category:Buildings and structures in Greece category:Politics of Greece category:Military of Greece category:Greek people category:Geography of Greece category:Communications in Greece category:Government of Greece category:Greek families category:Wikipedia books on Greece category:Time in Greece category:Greek culture category:Economy of Greece category:Greek society category:Greece-related lists category:History of Greece category:Greek law category:Education in Greece category:Science and technology in Greece category:Sport in Greece category:Geology of Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) | Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) | [N3/Turtle](#) | [JSON](#) | [XML](#)) | [OData](#) ([Atom](#) | [JSON](#)) | [Microdata](#) ([JSON](#) | [HTML](#)) | [JSON-LD](#) | [About](#)



This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Buildings and structures in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Buildings and structures in Greece Category:
skos:broader	category:Greece category:Greek architecture category:Buildings and structures by country
skos:prefLabel	Buildings and structures in Greece
is dcterms:subject of	dbpedia:Burgas–Alexandroupoli pipeline dbpedia:List of tallest buildings and structures in Greece
is skos:broader of	category:Airports in Greece category:Ancient Greek buildings and structures category:Places of worship in Greece category:Palaces in Greece category:Cemeteries in Greece category:Theatres in Greece category:Aqueducts in Greece category:Tunnels in Greece category:Castles in Greece category:Towers in Greece category:Lighthouses in Greece category:Museums in Greece category:Reservoirs in Greece category:Shopping malls in Greece category:Music venues in Greece category:Libraries in Greece category:Markets in Greece category:Buildings and structures in Greece by city category:Bridges in Greece category:Power stations in Greece category:Former buildings and structures of Greece category:Monuments and memorials in Greece category:Infrastructure in Greece category:Archaeological sites in Greece category:Sports venues in Greece category:Prisons in Greece category:Gates in Greece category:Astronomical observatories in Greece category:Houses in Greece category:Hotels in Greece category:Mines in Greece category:Oil refineries in Greece category:Railway stations in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Communications in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Communications in Greece
skos:broader	category:Greece
skos:prefLabel	Communications in Greece
is dcterms:subject of	dbpedia:gr dbpedia:List of postal codes in Greece dbpedia:List of people on stamps of Greece dbpedia:Postage stamps and postal history of Northern Epirus dbpedia:List of radio stations in Greece dbpedia:Greek Research and Technology Network dbpedia:Aphrodite 2 dbpedia:Fryctoria dbpedia:Critics.gr dbpedia:Patras wireless metropolitan network
is skos:broader of	category:Greek media category:Postage stamps of Greece category:Photography in Greece category:Telecommunications in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Economy of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Economy of Greece Category:
skos:broader	category:Greece category:Economies by country
skos:prefLabel	Economy of Greece
is dcterms:subject of	dbpedia:Economy of Greece dbpedia:Greek euro coins dbpedia:Hellenic Capital Market Commission dbpedia:Greek Steamship Company dbpedia:Bank of Greece dbpedia:FTSE/Athex 20 dbpedia:Athens Stock Exchange dbpedia:Greek Merchant Navy dbpedia:Fakelaki dbpedia:OPAP dbpedia:Euro gold and silver commemorative coins (Greece) dbpedia:International rankings of Greece dbpedia:Debtocracy
is skos:broader of	category:Greek billionaires category:Companies based in Athens category:Companies of Greece category:Energy in Greece category:Mining in Greece category:Retailing in Greece category:Taxation in Greece category:Tourism in Greece category:Economic history of Greece category:Greek businesspeople category:Agriculture in Greece category:Greek philanthropists category:Trade shows in Greece category:Economy of ancient Greece category:Revenue stamps of Greece category:Trade unions in Greece category:Economy of Athens category:Banking in Greece category:Coins of Greece category:Cars of Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Education in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Education in Greece
skos:broader	category:Greece category:Education in Southern Europe category:Education by country
skos:prefLabel	Education in Greece
is dcterms:subject of	dbpedia:IEK dbpedia:Education in Greece dbpedia:List of universities in Greece dbpedia:Diplom dbpedia:Polytechnic (Greece) dbpedia:Ministry of Education, Lifelong Learning and Religious Affairs (Greece) dbpedia:Apolytirion dbpedia:Frontistirio dbpedia:Article 16 of the Constitution of the Hellenic Republic dbpedia:TEI of Piraeus dbpedia:Academic grading in Greece dbpedia:Center for Hellenic Studies in Greece, Harvard University
is skos:broader of	category:Greek educationists category:Art schools in Greece category:Universities in Greece category:Museums in Greece category:Schools in Greece category:Libraries in Greece category:Education in ancient Greece category:Education in Athens category:People by educational institution in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Environment of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Environment of Greece
skos:broader	category:Greece category:Environment of the Mediterranean category:Environment by country category:Environment of Europe
skos:prefLabel	Environment of Greece
is dcterms:subject of	dbpedia:List of birds of Greece dbpedia:Balkan mixed forests dbpedia:Environmental issues in Greece dbpedia:Kalamiaris palm forest
is skos:broader of	category:Energy in Greece category:Fauna of Greece category:Environmental organizations based in Greece category:Biota of Greece category:Climate of Greece category:Greek botanists category:Conservation in Greece category:Flora of Greece category:Water supply and sanitation in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Geography of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Geography of Greece Category:
skos:broader	category:Greece category:Geography by country
skos:prefLabel	Geography of Greece
skos:related	category:Populated places in Greece
is dcterms:subject of	dbpedia:Thermopylae dbpedia:Necromanteion dbpedia:Thrace dbpedia:Epirus (region) dbpedia:Macedonia (Greece) dbpedia:Extreme points of Greece dbpedia:Pelagonia dbpedia:Artemisium dbpedia:Kato Dafni dbpedia:Scillus dbpedia:Nas (Ikaria) dbpedia:Plateia dbpedia:Regions of Greece dbpedia:Filiates dbpedia:Petrified forest of Lesbos dbpedia:Doroufi dbpedia:Makedonia Tembi dbpedia:Paralia Platanos dbpedia:Rizari dbpedia:Geography of Greece dbpedia:Amphiareion of Oropos dbpedia:Chech
is skos:broader of	category:Athens category:Thrace category:Epirus category:National parks of Greece category:Canals in Greece category:Chech category:Borders of Greece category:Oil fields in Greece category:Historical regions in Greece category:Subdivisions of Greece category:Reservoirs in Greece category:Parks in Greece category:Landforms of Greece category:Geography of Athens category:Geography of ancient Thrace category:Ports and harbours of Greece category:Macedonia category:Populated places in Greece category:Maps of Greece category:Natural disasters in Greece

[category:Forests of Greece](#)
[category:Geology of Greece](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Geology of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Geology of Greece
skos:broader	category:Greece category:Geography of Greece category:Geology of Europe category:Geology by country
skos:prefLabel	Geology of Greece
is dcterms:subject of	dbpedia:Hellenic arc dbpedia:Mediterranean Bauxite Province dbpedia:Verd antique
is skos:broader of	category:Mining in Greece category:Oil fields in Greece category:Earthquakes in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Government of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Government of Greece
skos:broader	category:Greece category:Politics of Greece category:Government by country
skos:prefLabel	Government of Greece
is dcterms:subject of	dbpedia:Cabinet of Greece dbpedia:Speaker of the Hellenic Parliament dbpedia:Greek wiretapping case 2004–2005 dbpedia:Minister for National Defence (Greece) dbpedia:Minister for the Press and the Media (Greece) dbpedia:Greek passport dbpedia:Minister for Macedonia–Thrace (Greece) dbpedia:General Secretariat for Macedonia and Thrace dbpedia:Minister for the Environment, Energy and Climate Change (Greece) dbpedia:Government Gazette (Greece) dbpedia:Government Council for Foreign Affairs and Defense dbpedia:Minister of State (Greece) dbpedia:Minister for Industry, Energy and Technology (Greece) dbpedia:Deputy Prime Minister of Greece
is skos:broader of	category:Government ministries in Greece category:Official residences in Greece category:Hellenic Parliament category:Taxation in Greece category:Subdivisions of Greece category:Government ministers of Greece category:Political office-holders in Greece category:Foreign relations of Greece category:Heads of state of Greece category:Presidency of the Hellenic Republic category:Orders, decorations, and medals of Greece category:National Agencies of Greece category:Lists of government ministers of Greece category:Cabinets of Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greece-related lists](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Greece-related lists
skos:broader	category:Greece category:Lists by country category:Europe-related lists
skos:prefLabel	Greece-related lists
is dcterms:subject of	dbpedia:List of Greek supermarket chains dbpedia:King of Athens dbpedia:List of universities in Greece dbpedia:List of museums in Greece dbpedia:Vehicle registration plates of Greece dbpedia:List of the prefectures of Greece by population density dbpedia:List of schools in Greece dbpedia:Speaker of the Hellenic Parliament dbpedia:List of newspapers in Greece dbpedia:Line of succession to the former Greek throne dbpedia:Peripheries of Greece dbpedia:Prefectures of Greece dbpedia:List of companies of Greece dbpedia:List of airports in Greece dbpedia:List of municipalities and communities in Greece dbpedia:List of postal codes in Greece dbpedia:List of hospitals in Greece dbpedia:List of people on stamps of Greece dbpedia:List of Greek films dbpedia:List of the prefectures of Greece by population dbpedia:List of Greek flags dbpedia:List of stoae dbpedia:List of islands of Greece dbpedia:List of Greeks dbpedia:List of Prime Ministers of Greece dbpedia:List of Kings of Sparta dbpedia:List of rivers of Greece dbpedia:Highways in Greece dbpedia:Telephone numbers in Greece dbpedia:List of Greek-language newspapers dbpedia:List of football clubs in Greece dbpedia:List of political parties in Greece dbpedia:List of archbishops of Athens dbpedia:National parks of Greece dbpedia:List of lakes in Greece dbpedia:List of current ships of the Hellenic Navy dbpedia:List of Archbishops of Crete dbpedia:List of the prefectures of Greece by area dbpedia:List of municipalities and communities of Greece by prefecture dbpedia:Companies listed on the Athens Stock Exchange dbpedia:List of prisons in Greece dbpedia:List of cities in Greece dbpedia:List of heads of state of Greece dbpedia:Peripheral units of Greece dbpedia:ISO 3166-2:GR dbpedia:List of caves in Greece dbpedia:List of mountains in Greece

[dbpedia:Regions of Greece](#)
[dbpedia:List of spa towns in Greece](#)
[dbpedia:Lists of rulers of Greece](#)
[dbpedia:List of mammals of Greece](#)
[dbpedia:List of banks in Greece](#)
[dbpedia:List of Greek language television channels](#)
[dbpedia:List of parliamentary constituencies of Greece](#)
[dbpedia:List of historic aircraft of the Hellenic Air Force](#)
[dbpedia:Outline of Greece](#)
[dbpedia:List of Roman Catholic dioceses in Greece](#)
[dbpedia:List of Greek think tanks](#)
[dbpedia:List of ports in Greece](#)
[dbpedia:List of diplomatic missions of Greece](#)
[dbpedia:List of birds of Greece](#)
[dbpedia:List of diplomatic missions in Greece](#)
[dbpedia:List of foreign football players in Super League Greece](#)
[dbpedia:List of wars involving Greece](#)
[dbpedia:List of football stadiums in Greece](#)
[dbpedia:List of Greek submissions for the Academy Award for Best Foreign Language Film](#)
[dbpedia:List of cabinets of Greece](#)
[dbpedia:List of Cretans](#)
[dbpedia:List of neighbourhoods in Patras](#)
[dbpedia:Former toponyms in Xanthi Prefecture](#)
[dbpedia:List of Greek countries and regions](#)
[dbpedia:List of tallest buildings and structures in Greece](#)
[dbpedia:Former toponyms in Florina Prefecture](#)
[dbpedia:TEI of Piraeus](#)
[dbpedia:List of Foreign Archaeological Institutes in Greece](#)
[dbpedia:Former toponyms in Pella Prefecture](#)
[dbpedia:List of airlines of Greece](#)
[dbpedia:Index of Greece-related articles](#)
[dbpedia:List of football clubs in Greece by major honours won](#)
[dbpedia:List of twin towns and sister cities in Greece](#)
[dbpedia:List of Greek Armenians](#)
[dbpedia:List of massacres in Greece](#)
[dbpedia:List of non-marine molluscs of Greece](#)
[dbpedia:List of political families in Greece](#)
[dbpedia:List of Panathinaikos F.C. presidents](#)
[dbpedia:List of members of the Academy of Athens](#)
[dbpedia:List of freshwater fishes of Greece](#)
[dbpedia:List of Greek musical artists](#)
[dbpedia:Visa requirements for Greek citizens](#)
[dbpedia:List of kings of Argos](#)
[dbpedia:List of earthquakes in Greece](#)
[dbpedia:List of foreign football players in the Greek Superleague 2010–11](#)
[dbpedia:List of municipalities of Greece \(2011\)](#)

is [skos:broader](#) of [category:Lists of populated places in Greece](#)
[category:Ancient Greece-related lists](#)
[category:Lists of Greek people](#)
[category:Lists of Greek films by decade](#)
[category:Lists of government ministers of Greece](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greek culture](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Greek culture Category:
skos:broader	category:Greece category:European culture category:Culture by nationality
skos:prefLabel	Greek culture
is dcterms:subject of	dbpedia:Acropolis of Athens dbpedia:White Tower of Thessaloniki dbpedia:Twelve Olympians dbpedia:Ancient Greek clubs dbpedia:Famous Macedonia dbpedia:Clean Monday dbpedia:Thessaloniki International Film Festival dbpedia:Petros (pelican) dbpedia:Rouketopolemos dbpedia:Museum of the Center for the Acropolis Studies dbpedia:Trata dbpedia:National Hellenic Museum dbpedia:Stephen of Alexandria dbpedia:Apollon Theatre (Patras) dbpedia:Culture of Greece dbpedia:Archaeological Museum of Thessaloniki dbpedia:Cinema of Greece dbpedia:Greek festival dbpedia:Heraklion Archaeological Museum dbpedia:Artforum Culture Foundation dbpedia:Acropolis dbpedia:Leonidas I dbpedia:Greek War of Independence dbpedia:Epidaurus dbpedia:Cleisthenes dbpedia:Hora (dance) dbpedia:Zeibekiko dbpedia:Names of the Greeks dbpedia:Byzantine art dbpedia:Kallikantzaros dbpedia:Propylaea dbpedia:Macedonian Renaissance dbpedia:Theatre of Dionysus dbpedia:Cretan lyra dbpedia:Greektown dbpedia:Benaki Museum dbpedia:Stoivadeion dbpedia:Laiiko dbpedia:Tsakonikos dbpedia:Greek name dbpedia:Olympus Festival dbpedia:Merry Crisis dbpedia:Endeka Kozanis dbpedia:Kourbania dbpedia:Old Acropolis Museum dbpedia:Hercules

[dbpedia:Temple of Olympian Zeus, Athens](#)
[dbpedia:Pericles](#)
[dbpedia:Caryatid](#)
[dbpedia:Sirtaki](#)
[dbpedia:Hasapiko](#)
[dbpedia:Ministry of Culture and Tourism \(Greece\)](#)
[dbpedia:Monastery of the Cross](#)
[dbpedia:Byzantine Greeks](#)
[dbpedia:Ikariotikos](#)
[dbpedia:Diotima \(magazine\)](#)
[dbpedia:Kombolói](#)
[dbpedia:Ohi Day](#)
[dbpedia:Rebetiko](#)
[dbpedia:National Theatre of Greece](#)
[dbpedia:Mar Saba](#)
[dbpedia:Hellenic studies](#)
[dbpedia:Macedonian art \(Byzantine\)](#)
[dbpedia:Archaeological Museum of Piraeus](#)
[dbpedia:University of Constantinople](#)
[dbpedia:Greek East and Latin West](#)
[dbpedia:Fakelaki](#)
[dbpedia:Saitopolemos](#)
[dbpedia:Kalamatianos](#)
[dbpedia:Anastenaria](#)
[dbpedia:Acropolis Museum](#)
[dbpedia:Mangas](#)
[dbpedia:Greek dances](#)
[dbpedia:Lydia Lithos Dance Theatre](#)
[dbpedia:Anglo-Hellenic League](#)
[dbpedia:Festivals of Thessaloniki](#)
[dbpedia:Museum of Ancient Greek, Byzantine and Post-Byzantine Instruments](#)
[dbpedia:Center for the Greek language](#)
[dbpedia:Ái Georgis](#)
[dbpedia:Hellenic Foundation for Culture](#)
[dbpedia:Philotimo](#)
[dbpedia:Folk Art Museum of Acharnes](#)
[dbpedia:Delphi Archaeological Museum](#)
[dbpedia:Kalamatianó](#)
[dbpedia:Loulouvikos](#)
[dbpedia:Name of Greece](#)
[dbpedia:Archaeological Museum of Nafplion](#)
[dbpedia:Kapodistrias Museum](#)
[dbpedia:Archaeological Museum of Ioannina](#)
[dbpedia:Greek musical instruments](#)
[dbpedia:Parnassos Literary Society](#)
[dbpedia:Smoking in Greece](#)
[dbpedia:OPA! Day](#)

is [skos:broader](#) of

[category:Languages of Greece](#)
[category:Pontus](#)
[category:Modern Greek studies](#)
[category:Photography in Greece](#)
[category:Ancient Greek culture](#)
[category:National symbols of Greece](#)
[category:Flags of Greece](#)
[category:Greek American culture](#)
[category:Museums in Greece](#)
[category:Arts in Greece](#)
[category:Festivals in Greece](#)
[category:Libraries in Greece](#)
[category:Ethnic groups in Greece](#)
[category:Greek literature](#)
[category:Greek dances](#)

[category:Greek folklore](#)
[category:Greek Orthodoxy](#)
[category:Greek clothing](#)
[category:Culture in Athens](#)
[category:Orders, decorations, and medals of Greece](#)
[category:Events in Greece](#)
[category:Works by Greek people](#)
[category:Entertainment in Greece](#)
[category:Greek brands](#)
[category:Greek awards](#)
[category:Greek cuisine](#)
[category:Greece in fiction](#)
[category:Culture of Crete](#)
[category:Religion in Greece](#)
[category:Sport in Greece](#)
[category:Greek fashion](#)
[category:Greek coats of arms](#)
[category:Burials in Greece](#)
[category:Animal breeds originating in Greece](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greek families](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Greek families
skos:broader	category:Greece category:Families by nationality
skos:prefLabel	Greek families
is dcterms:subject of	dbpedia:Mourousis family dbpedia:Zaimis family dbpedia:Manos family dbpedia:Ypsilantis dbpedia:Gattilusi dbpedia:Levidis family dbpedia:Racoviță dbpedia:Flessas family dbpedia:Fotilas dbpedia:Caradja dbpedia:Gerontas dbpedia:Langoura dbpedia:Petimezas dbpedia:Kanakaris dbpedia:Kalamogdartis family dbpedia:Kallergis family dbpedia:Karavias dbpedia:Skouzes family dbpedia:Tzavaras dbpedia:Lemos family
is skos:broader of	category:Ancient Greek families category:Livanos family category:Relatives of Socrates category:Miaoulis family category:Mavrocordatos family category:Avgerinos family category:Political families of Greece category:Callimachi family category:Chalkokondyles family category:Niarchos family category:Vilaetis family category:Greek noble families category:Ypsilantis family category:Caradja family category:Byzantine families category:Soutzos family category:Onassis family category:Goulandris family category:Cantacuzino family category:Ghica family category:Racoviță family category:Petimeza/Petmeza family category:Ephrussi family category:Mavromichalis family category:Maleinos family

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greek law](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Greek law
skos:broader	category:Greece category:Law by country
skos:prefLabel	Greek law
is dcterms:subject of	dbpedia:Copyright law of Greece dbpedia:Human rights in Greece dbpedia:Right of return dbpedia:Constitution of Greece dbpedia:Law 3037/2002 dbpedia:Law 4000/1958 dbpedia:Idionymon dbpedia:Greek nationality law dbpedia:Government Gazette (Greece) dbpedia:Abortion in Greece dbpedia:Capital punishment in Greece
is skos:broader of	category:Judicial system of Greece category:Law enforcement in Greece category:LGBT rights in Greece category:Hellenic Parliament category:Taxation in Greece category:Referendums in Greece category:Greek judges category:Greek courts category:Greek lawyers category:Crime in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greek music](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Greek music Category:
skos:broader	category:Greece category:Southeastern European music category:Music by nationality category:Arts in Greece category:Southern European music
skos:prefLabel	Greek music
is dcterms:subject of	dbpedia:Greek hip hop dbpedia:Lyre dbpedia:Zurna dbpedia:Byzantine music dbpedia:Dimitris Voyatzis dbpedia:Famous Macedonia dbpedia:Diatonic and chromatic dbpedia:Trata dbpedia:Timeline of Rebetika dbpedia:Greek folk music dbpedia:Hymn to Liberty dbpedia:Makedonia (dance) dbpedia:Music of Macedonia (Greece) dbpedia:Pentozali dbpedia:Cretan lyra dbpedia:Greek rock dbpedia:Ionian School (music) dbpedia:Tromakton dbpedia:Eurovision Song Contest 2006 dbpedia:Leventikos dbpedia:Laïko dbpedia:Zonaradiko dbpedia:Greece in the Junior Eurovision Song Contest dbpedia:Athens Conservatoire dbpedia:National Conservatoire (Greece) dbpedia:Hellenic Conservatory dbpedia:The Last Drive dbpedia:Bouzouki dbpedia:Music of Greece dbpedia:Sirtaki dbpedia:Hasapiko dbpedia:Antikrystos dbpedia:Canon (hymnography) dbpedia:Ikariotikos dbpedia:Misirlou dbpedia:Rebetiko dbpedia:Music of Crete dbpedia:Music of Thrace dbpedia:Low Bap dbpedia:Guardians of Hellenism dbpedia:MTV Europe Music Award for Best Greek Act

[dbpedia:Tsestos](#)
[dbpedia:Kalamatianos](#)
[dbpedia:Makedonikos antikristos](#)
[dbpedia:Tzouras](#)
[dbpedia:Polyphonic song of Epirus](#)
[dbpedia:Endelekheia](#)
[dbpedia:Souravli](#)
[dbpedia:Octave species](#)
[dbpedia:Greek New Wave](#)
[dbpedia:Tagsim](#)
[dbpedia:Kapitan Louka](#)
[dbpedia:Syrto](#)
[dbpedia:Museum of Ancient Greek, Byzantine and Post-Byzantine Instruments](#)
[dbpedia:The Dead Brother's Song](#)
[dbpedia:Hellenic Electroacoustic Music Composers Association](#)
[dbpedia:Manolis Chiotis](#)
[dbpedia:Music of Epirus \(Greece\)](#)
[dbpedia:Tambouras](#)
[dbpedia:Skyladiko](#)
[dbpedia:Mihanikos](#)
[dbpedia:Kalamatianó](#)
[dbpedia:Loulouvikos](#)
[dbpedia:Amoebaeon singing](#)
[dbpedia:Lavta](#)
[dbpedia:Baglamas](#)
[dbpedia:Askomandoura](#)
[dbpedia:Byzantine lyra](#)
[dbpedia:National Theatre of Greece Drama School](#)
[dbpedia:Greek musical instruments](#)
[dbpedia:Karamuza](#)
[dbpedia:Thaboura](#)
[dbpedia:Lalitsa](#)
[dbpedia:Mantura](#)
[dbpedia:Karantouzeni](#)
[dbpedia:Cochilia](#)
[dbpedia:Zervos](#)
[dbpedia:Diplos Horos](#)
[dbpedia:Zilia \(musical instruments\)](#)
[dbpedia:Toubeleki](#)
[dbpedia:Koudounia](#)
[dbpedia:Kamilierikos Choros](#)
[dbpedia:Trigono \(musical instrument\)](#)

is [skos:broader](#) of

[category:Greek record labels](#)
[category:Greek songs](#)
[category:Greek record charts](#)
[category:Albums by Greek artists](#)
[category:Greek musical instruments](#)
[category:Number-one singles in Greece](#)
[category:Music venues in Greece](#)
[category:Discographies of Greek artists](#)
[category:Greek dances](#)
[category:Greek musicologists](#)
[category:Greek musicians](#)
[category:Rebetiko](#)
[category:Greece in the Eurovision Song Contest](#)
[category:Greek hip hop](#)
[category:Greek musical groups](#)
[category:Greek singers](#)
[category:Greek music managers](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greek people](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Greek people Category:
skos:broader	category:Greece category:European people category:Indo-European peoples category:People by nationality category:Ethnic groups in the Balkans
skos:prefLabel	Greek people
skos:related	category:Greek Cypriot people category:Greek Turkish people
is dcterms:subject of	dbpedia:Solon dbpedia:Phanariotes dbpedia:Maria Damanaki dbpedia:Stavros Fasoulas dbpedia:Demographic history of modern Greece dbpedia:Dimitrios Miaoulis dbpedia:Rallou Manou dbpedia:Tryfon Tolides dbpedia:Greeks dbpedia:Epirus (region) dbpedia:Perses dbpedia:Klepht dbpedia:Costas Azariadis dbpedia:Demographic history of Greece dbpedia:Georgios Karaiskakis dbpedia:George of Antioch dbpedia:Praxiteles dbpedia:Leonidas I dbpedia:List of Greeks dbpedia:Cleisthenes dbpedia:Names of the Greeks dbpedia:Tsakonians dbpedia:Konstantinos Miaris dbpedia:Sarakatsani dbpedia:Levidis family dbpedia:Stratioti dbpedia:Pericles dbpedia:Population of the Byzantine Empire dbpedia:Byzantine Greeks dbpedia:Macedonians (Greeks) dbpedia:Ioannis Kapodistrias dbpedia:Karamanlides dbpedia:Maniots dbpedia:Nicholas Lambrinides dbpedia:Sfakians dbpedia:Mimis Androulakis dbpedia:Greeks in Albania dbpedia:John Argyris dbpedia:Greek refugees

[dbpedia:Great_Greeks](#)
[dbpedia:Adonis_Kyrou](#)
[dbpedia:Mandritsa](#)
[dbpedia:Nicolas_Ambraseys](#)
[dbpedia:Souliotes](#)
[dbpedia:Dimitri_Bertsekas](#)
[dbpedia:Mihail_Zervos](#)
[dbpedia:Maggira_Sisters](#)
[dbpedia:List_of_Greeks_who_were_born_outside_modern_Greece](#)

is [skos:broader](#) of

[category:Greek_billionaires](#)
[category:Greek_Nobel_laureates](#)
[category:People_by_city_or_town_in_Greece](#)
[category:Greek_women](#)
[category:WikiProject_Greece_people_articles](#)
[category:Greek_crime_victims](#)
[category:Categories_named_after_Greek_people](#)
[category:Greek_vegetarians](#)
[category:Greek_Renaissance_humanists](#)
[category:Lists_of_Greek_people](#)
[category:Byzantine_people](#)
[category:Greek_people_by_century](#)
[category:Ottoman_Greeks](#)
[category:Greek_people_by_war](#)
[category:Images_of_Greek_people](#)
[category:Ancient_Greeks](#)
[category:LGBT_people_from_Greece](#)
[category:Greek_prisoners_and_detainees](#)
[category:Greek_diaspora](#)
[category:Greek_people_by_occupation](#)
[category:Greek_people_by_ethnic_or_national_origin](#)
[category:Greek_nobility](#)
[category:Naturalized_citizens_of_Greece](#)
[category:People_by_prefecture_in_Greece](#)
[category:Greek_children](#)
[category:Greek centenarians](#)
[category:Fictional_Greek_people](#)
[category:Arvanites](#)
[category:Greek_Macedonians](#)
[category:Greek_royalty](#)
[category:Greek_people_by_religion](#)
[category:Greek_people_by_political_orientation](#)
[category:Immigrants_to_Greece](#)
[category:People_by_periphery_in_Greece](#)
[category:Greek_hermits](#)
[category:Greek_people_with_disabilities](#)
[category:People_by_educational_institution_in_Greece](#)
[category:People_by_island_in_Greece](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Greek society](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Greek society
skos:broader	category:Greece category:Society by nationality
skos:prefLabel	Greek society
is dcterms:subject of	dbpedia:Demographic history of modern Greece dbpedia:Minorities in Greece dbpedia:Human rights in Greece dbpedia:Demographic history of Greece dbpedia:Greek War of Independence dbpedia:Hellenization dbpedia:Muslim minority of Greece dbpedia:Demographics of Greece dbpedia:Population of the Byzantine Empire dbpedia:Fakelaki dbpedia:Greeks in Albania dbpedia:Greek refugees dbpedia:Armenians in Greece dbpedia:Iraqis in Greece dbpedia:Feminism in Greece dbpedia:International rankings of Greece dbpedia:Scuola dei Greci
is skos:broader of	category:Languages of Greece category:Ancient Greek society category:Demographics of Greece category:Drugs in Greece category:Ethnic groups in Greece category:Greek nobility category:Greek activists category:Orders, decorations, and medals of Greece category:Immigration to Greece category:Death in Greece category:Antisemitism in Greece category:Greek awards category:Organizations based in Greece category:Religion in Greece category:Crime in Greece category:Greek coats of arms category:Human rights in Greece category:LGBT in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Health in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Health in Greece
skos:broader	category:Greece category:Health by country category:Health in Europe
skos:prefLabel	Health in Greece
is dcterms:subject of	dbpedia:Hellenic Centre for Diseases Control and Prevention dbpedia:Greek Alliance of Rare Diseases dbpedia:Obesity in Greece dbpedia:Smoking in Greece
is skos:broader of	category:Drugs in Greece category:Greek physicians category:Disasters in Greece category:Healthcare in Greece category:Death in Greece category:Water supply and sanitation in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [History of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	History of Greece Category:
skos:broader	category:Greece category:History of the Balkans by country category:History of Europe category:History by country
skos:prefLabel	History of Greece
is dcterms:subject of	dbpedia:History of Athens dbpedia:History of Greece dbpedia:Praetorian prefecture of Illyricum dbpedia:Jireček Line dbpedia:List of ancient tribes in Thrace and Dacia dbpedia:Thrace dbpedia:History of Crete dbpedia:Zagori dbpedia:Illyrians dbpedia:Names of the Greeks dbpedia:1920 in Greece dbpedia:Hellenization dbpedia:Epidamnos dbpedia:Phanagoria dbpedia:Illyria dbpedia:Demographic history of Macedonia dbpedia:Byzantine Greece dbpedia:Republic of Venice dbpedia:Byzantine Greeks dbpedia:Cicones dbpedia:Lists of rulers of Greece dbpedia:Panhellenism dbpedia:Griko people dbpedia:Northern Epirote Declaration of Independence dbpedia:Name of Greece dbpedia:List of Greek countries and regions dbpedia:List of ancient tribes in Illyria
is skos:broader of	category:Illyria category:Greek colonies in Chalcidice category:History of Greece by location category:Military history of Greece category:Protests in Greece category:Greek colonies in Macedonia category:Jewish Byzantine history category:Economic history of Greece category:Historical regions in Greece category:LGBT history in Greece category:Byzantine Empire category:History of rail transport in Greece category:Greek diaspora category:Greek colonies in Illyria category:Political history of Greece

[category:History museums in Greece](#)
[category:Greek rebellions](#)
[category:Disasters in Greece](#)
[category:Massacres in Greece](#)
[category:Former buildings and structures of Greece](#)
[category:Greek colonies in Pieria](#)
[category:Jewish Greek history](#)
[category:Archaeology of Greece](#)
[category:Maps of the history of Greece](#)
[category:Maritime history of Greece](#)
[category:History of Greece by period](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Images of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Images of Greece Category:
skos:broader	category:Greece category:Images by country category:Images of Europe
skos:prefLabel	Images of Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Military of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Military of Greece Category:
skos:broader	category:Greece category:Military by country
skos:prefLabel	Military of Greece
is dcterms:subject of	dbpedia:Minister for National Defence (Greece) dbpedia:Military of Greece dbpedia:Conscription in Greece dbpedia:Ministry of National Defence (Greece) dbpedia:Government Council for Foreign Affairs and Defense dbpedia:Balkan Battle Group dbpedia:Hellenic National Defense General Staff dbpedia:International rankings of Greece
is skos:broader of	category:Hellenic Air Force category:Hellenic Army category:Hellenic Navy category:Military history of Greece category:Hellenic Coast Guard category:Greek military personnel category:Military ranks of Greece category:Military awards and decorations of Greece category:Military installations of Greece category:Defence companies of Greece category:Greek military writers category:Military units and formations of Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Politics of Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	Template:Pathnav Template:Pathnav Template:Commonscat
rdf:type	skos:Concept
rdfs:comment	Template:Pathnav Template:Pathnav Template:Commonscat
rdfs:label	Politics of Greece Category:
skos:broader	category:Greece category:Politics of Europe category:Politics by country
skos:prefLabel	Politics of Greece
is dcterms:subject of	dbpedia:Prime Minister of Greece dbpedia:Parataxis (politics) dbpedia:Revolutionary Nuclei dbpedia:Chamber of Accounts (Greece) dbpedia:Nuclear energy in Greece dbpedia:2010–2011 Greek protests dbpedia:Politics of Greece dbpedia:Macedonia (Greece) dbpedia:Greek Financial Audit, 2004 dbpedia:Supreme Special Court (Greece) dbpedia:Idionymon dbpedia:Athens Declaration of the European Left dbpedia:Council of State (Greece) dbpedia:Liberalism in Greece dbpedia:Aegean dispute dbpedia:Greek Atomic Energy Commission dbpedia:Greek nationalism dbpedia:Evangelos Kofos dbpedia:Presidium of the Hellenic Parliament dbpedia:2008 Greek riots dbpedia:Greek Constitution of 1911 dbpedia:Parliamentary Committees (Greece) dbpedia:CIA activities in Greece dbpedia:Rebel Sect dbpedia:Archeio-Marxism dbpedia:List of cabinets of Greece dbpedia:Conference of Presidents (Greece) dbpedia:Ethniki Etaireia dbpedia:International rankings of Greece dbpedia:Obesity in Greece
is skos:broader of	category:Terrorism in Greece category:Greek politicians category:Groups that resisted the Greek military junta of 1967–1974 category:LGBT rights in Greece category:Protests in Greece category:Hellenic Parliament category:Political families of Greece category:Political advocacy groups in Greece category:Foreign relations of Greece category:Political history of Greece category:Constitution of Greece category:Presidency of the Hellenic Republic category:Government of Greece

[category:Political scandals in Greece](#)
[category:Republicanism in Greece](#)
[category:Political parties in Greece](#)
[category:Greek people by political orientation](#)
[category:Human rights in Greece](#)
[category:Greek political scientists](#)
[category:Communism in Greece](#)

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in:
[CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Science and technology in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Science and technology in Greece
skos:broader	category:Greece category:Science and technology in Europe category:Science and technology by country
skos:prefLabel	Science and technology in Greece
is dcterms:subject of	dbpedia:List of universities in Greece dbpedia:Hellenic Journal of Geosciences dbpedia:Technical Chamber of Greece dbpedia:Greek Wikipedia dbpedia:Alexander Fleming Biomedical Sciences Research Center dbpedia:Foundation for Research dbpedia:Ionian University dbpedia:Thessaloniki Science Center and Technology Museum dbpedia:National Technical University of Athens dbpedia:Technical University of Crete dbpedia:List of research institutes in Greece dbpedia:CERETETH dbpedia:Science Park Zakynthos dbpedia:TEI of Piraeus
is skos:broader of	category:Ancient Greek science category:Greek science writers category:Greek inventions category:Greek scientists category:Greek inventors category:Greek engineers category:Power stations in Greece category:Research institutes in Greece category:Greek technology writers category:Nuclear technology in Greece category:Academy of Athens category:Astronomical observatories in Greece category:Internet in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Sport in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
dbpedia-owl:abstract	
rdf:type	skos:Concept
rdfs:comment	
rdfs:label	Sport in Greece Category:
skos:broader	category:Greece category:Entertainment in Greece category:Greek culture category:Sport in Europe category:Sport by country
skos:prefLabel	Sport in Greece
is dcterms:subject of	dbpedia:Spartathlon dbpedia:Gate 13 dbpedia:Sport in Greece dbpedia:Balkan Games
is skos:broader of	category:Greek sportspeople category:Sport in Crete category:Sports teams in Greece category:Athletics in ancient Greece category:Sport in Messenia category:Sports competitions in Greece category:Sports venues in Greece category:Sport in Greece by city category:Sports governing bodies in Greece category:Sport in Greece by sport

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Time in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Time in Greece
skos:broader	category:Greece category:Time by country
skos:prefLabel	Time in Greece
is dcterms:subject of	dbpedia:Date and time notation in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Transport in Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Transport in Greece Category:
skos:broader	category:Greece category:Transport in Europe category:Transport by country
skos:prefLabel	Transport in Greece
is dcterms:subject of	dbpedia:Vehicle registration plates of Greece dbpedia:Aktio–Preveza Undersea Tunnel dbpedia:Transport in Greece dbpedia:Eleftheriou Venizelou Street dbpedia:Thessaloniki Urban Transport Organization dbpedia:Ministry of Infrastructure, Transport and Networks
is skos:broader of	category:Canals in Greece category:Tunnels in Greece category:Cycling in Greece category:Transport in Athens category:Roads in Greece category:Rail transport in Greece category:Water transport in Greece category:Merchant ships of Greece category:Transport in Ilia category:History of transport in Greece category:Aviation in Greece category:Bridges in Greece category:Public transport in Greece category:Greece transport templates category:Ports and harbours of Greece category:Transport disasters in Greece category:Transport infrastructure in Greece category:Cable cars in Greece

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | [RDF](#) ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | [OData](#) ([Atom](#) [JSON](#)) | [Microdata](#) ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

About: [Wikipedia books on Greece](#)

An Entity of Type : [Concept](#), from Named Graph : <http://dbpedia.org/>, within Data Space : dbpedia.org

Property	Value
rdf:type	skos:Concept
rdfs:label	Wikipedia books on Greece
skos:broader	category:Greece category:Wikipedia books on countries category:Wikipedia books on Europe
skos:prefLabel	Wikipedia books on Greece
is dcterms:subject of	dbpedia:Book:Pergamon and its History dbpedia:Book:Greek Mythology - 2 : Heroes and Creatures dbpedia:Book:Greek Mythology - 1 : Deities

Browse using: [OpenLink Data Explorer](#) | [Zitgist Data Viewer](#) | [Marbles](#) | [DISCO](#) | [Tabulator](#) Raw Data in: [CSV](#) | RDF ([N-Triples](#) [N3/Turtle](#) [JSON](#) [XML](#)) | OData ([Atom](#) [JSON](#)) | Microdata ([JSON](#) [HTML](#)) | [JSON-LD](#) [About](#)

This content was extracted from [Wikipedia](#) and is licensed under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#)

11. APPENDIX B – The code

The main part of the code is presented in this Appendix. Other parts of the code used on the website and not relevant to the thesis are not included.

```
*****
<html>
<head>
<script type="text/javascript"
src="http://maps.googleapis.com/maps/api/js?sensor=false"></script>
<script type="text/javascript" src="js/infowindowcss.js"></script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="refresh" content="60">
<STYLE TYPE="text/css">
<!--
A:link, A:visited { text-decoration: none }
A:hover { text-decoration: underline; color: none }
-->
</STYLE>

<BODY BGCOLOR="#000000"
      TEXT="#DC143C"
      LINK="#ffffff"
      ALINK="#58538A"
      VLINK="#C7C7C7"
      background="images/grMap2.jpg"
      MARGINWIDTH="0"
      MARGINHEIGHT="0"
      LEFTMARGIN="0"
      TOPMARGIN="0">

<div ALIGN=right>
<table border=0 CELLSPACING=0 CELLPADDING=10 WIDTH="300" HEIGHT="45">
  <tr>
    <td>

      <hr color="#ffffff">
      <center><font size="1" face="Verdana">
        <a href="index.php">Home</A>
        <a href="aboutUs.php">About us</A>
        <a href="videos.php">Videos</A>
        <a href="instructions.php">Instructions</A>
      </font></center>
    </td>
  </tr>
</table>
</div>

<?php
//save the user's preferences from the index.php page
$badgeID=$_GET['badgeID'];
$area=$_GET['area'];
$limPlaces=$_GET['limPlaces'];
//Uses the input user's Badge ID to find the location of the user.
$url="http://www.google.com/latitude/apps/badge/api?user=".$badgeID."&type=js
on" ;
// We get the content
```

```

$content = file_get_contents( $url );
// We convert the JSON to an object
$json = json_decode( $content );
//If the ID is wrong or the Google latitude doesnt have any information
if (empty($json->features))
    header('location: oops.php');
// retrieving the coordinates and other data of the user
$coord = $json->features[0]->geometry->coordinates;
$latitude=$coord[1];
$longitude=$coord[0];
$timeStamp = $json->features[0]->properties->timeStamp;
$whereAreU = $json->features[0]->properties->reverseGeocode;

?>

<?php function getNearBy($myLat,$myLong,$area2C,$lim2Show)
// function that returns the spots worth seeing nearBy the user
{ $format = 'json';

// The SPARQL Query
$query =
    "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

    SELECT DISTINCT ?subject (sql:SAMPLE(?subject) AS ?sbjct)
(sql:SAMPLE(?lat) AS ?lat) (sql:SAMPLE(?long) AS ?long) (sql:SAMPLE(?label)
AS ?label) (sql:SAMPLE(?abstract) AS ?abstract)

    WHERE {
        ?subject      geo:lat      ?lat;
                      geo:long     ?long;
                      rdfs:label    ?label;
                      dbo:abstract  ?abstract.

        { ?subject      dcterms:subject category:Greek_culture. }

    UNION

        {
            ?_category skos:broader category:Greek_culture .
            ?subject dcterms:subject ?_category .
        }

    UNION

        {
            ?_category skos:broader [skos:broader
category:Greek_culture] .
            ?subject dcterms:subject ?_category .
        }

    FILTER(
        xsd:float(?lat)-xsd:float(".$myLat.")<=$area2C    &&
        xsd:float(".$myLat.")-xsd:float(?lat)<=$area2C    &&
        xsd:float(?long)-xsd:float(".$myLong.")<=$area2C  &&
        xsd:float(".$myLong.")-xsd:float(?long)<=$area2C  &&
        lang(?abstract)='en' &&
        lang(?label)='en' ).

```

```

        } GROUP BY ?subject LIMIT (".$lim2Show.");

$searchUrl = 'http://dbpedia.org/sparql?'
    . 'query=' . urlencode($query)
    . '&format=' . $format;

    return $searchUrl;}
?>

<?php function request($url){
    // is curl installed?
    if (!function_exists('curl_init')){
        die('CURL is not installed!'); }
    // get curl handle
    $ch= curl_init();
    // set request url
    curl_setopt($ch,
        CURLOPT_URL,
        $url);
    // return response, don't print/echo
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    $response = curl_exec($ch);
    curl_close($ch);
    return $response;}
?>

<?php
//running the SPARQL Query
$myLat = $latitude;
$myLong = $longitude;
$area2C = $area/113;
$lim2Show = $limPlaces;
$requestURL = getNearBy($myLat,$myLong,$area2C,$lim2Show);
$responseArray = json_decode (
    request($requestURL),
    true);
?>
<title> Worth 2C </title>

<font size="5" face="Verdana" color="white" > <strong>worth2C Places Around
YOU </font>
<script type="text/javascript">
// function that initialize map
function initialize(lat,lng) {

// user's coordinates according to Google Latitude Badge
var myLatLng = new google.maps.LatLng(lat,lng);

// Map's Options
var myOptions = {
    zoom: 12,
    center: myLatLng,
    mapTypeId: google.maps.MapTypeId.ROADMAP };

var map = new google.maps.Map(document.getElementById('map_canvas'),
myOptions);

//Initialize User's Marker and Infobox
var markerIconA = new google.maps.MarkerImage(
    "images/footprint.png",
    new google.maps.Size(32,37),

```

```

        new google.maps.Point(0,0),
        new google.maps.Point(16,37) );
var markerA = new google.maps.Marker({
    position: myLatLng,
    icon: markerIconA,
    title:"You."});
markerA.setMap(map);

var infowindowA = new google.maps.InfoWindow({
    content: "You are...HERE!" });
google.maps.event.addListener(markerA, 'click', function() {
    infowindowA.open(map,markerA);});

// Using SPARQL's Results
var locations = [];
locations = [
    <?php for($i=0; $i<$lim2Show; $i++) { ?>
        ['<?php echo substr((str_replace('"','\'',@$responseArray
["results"]["bindings"][$i]["abstract"]["value"]),0,1200); ?>',
        <?php echo @$responseArray ["results"]["bindings"][$i]["lat"]["value"]
; ?>,
        <?php echo @$responseArray
["results"]["bindings"][$i]["long"]["value"]; ?>,
        '<?php echo str_replace('"','\'',@$responseArray
["results"]["bindings"][$i]["label"]["value"]); ?>' ],
        <?php ?> ]

//Initialize Worth2C Places' Markers and Infowindows
var markerIconB = new google.maps.MarkerImage(
    "images/museum-historical.png",
    new google.maps.Size(32,37),
    new google.maps.Point(0,0),
    new google.maps.Point(16,37) );
var infoBubble = new InfoBubble({
    map : map,
    maxWidth : 400,
    minWidth : 150,
    maxHeight : 300,
    minHeight : 50,
    shadowStyle : 1,
    padding : 10,
    backgroundColor : '#fdead0',
    borderRadius : 8,
    arrowSize : 15,
    borderWidth : 5,
    borderColor : '#ff8000',
    disableAutoPan : false,
    hideCloseButton : false,
    arrowPosition : 50,
    arrowStyle : 0
});

//Putting the Worth2C's markers on the map
var i;
for (i = 0; i < locations.length; i++) {
    markerB = new google.maps.Marker({
        position: new google.maps.LatLng(locations[i][1], locations[i][2]),
        map: map,
        icon: markerIconB,
        title:locations[i][3] });

```

```

        google.maps.event.addListener(markerB, 'click', (function(markerB, i)
{
    return function()
    {infoBubble.setContent(locations[i][0]);
      infoBubble.open(map, markerB); }
    }) (markerB, i)); } }

</script>

<script type="text/javascript">
function startMap() {
var lat = "<?php echo $latitude; ?>";
var lng = "<?php echo $longitude; ?>";
initialize(lat,lng);
}
</script>
</head>

<body >
<table border=0 CELLSPACING=0 CELLPADDING=10 WIDTH="100%">
    <tr>
        <td WIDTH = "70%">
            <script>
                window.onload = startMap;
            </script>
            <div id="map_canvas" style="width:800px;height:500px" ></div>
        </td>
        <td WIDTH = "30%" align="justify">
            <font size="3" face="Verdana" color = "white" ><strong><b> <?php echo
"According to Google Latitude you are in:<br>".$whereAreU; ?> <br><br>
            <?php echo "Your current coordinates are: <br> Latitude:
".$latitude." <br>Longitude: ".$longitude ; ?> </font>
        </td>
    </tr>

<div ALIGN=right>
    <p style=" position: absolute; bottom: 0; left: 0; width: 100%; color:
white; text-align: center;">
        Copyright (c) Maria Kommata - Vasilios Gongolidis <br>
        background photo Copyright (c) <a
href="http://www.flickr.com/photos/magisstra">Paolo Nespoli</A> </p>
</div>
</body>
</html>

```


12. Bibliography-References

- [1] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*, The MIT Press, 2008.
- [2] E. Chamopoulou, E. Katsaounou and M. Noti, *Educational material development for the Semantic Web technologies-Thesis*, National Technical University of Athens, University of Piraeus, 2011.
- [3] DBpedia. [Online]. Available: <http://dbpedia.org/About>.
- [4] B. DuCharme, *Learning SPARQL*, O'Reilly, 2011.
- [5] "Google Latitude," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Google_Latitude.
- [6] "LOD2-Creating Knowledge out of Interlinked Data," EU 7th Framework Programme, [Online]. Available: <http://lod2.eu/Welcome.html>.
- [7] E. M. L. (XML), W3C, [Online]. Available: <http://www.w3.org/XML/>.
- [8] "Document Type Definition (DTD)," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Document_Type_Definition.
- [9] "XML Schema Part 0: Primer Second Edition," W3C, [Online]. Available: <http://www.w3.org/TR/xmlschema-0/>.
- [10] "The XML FAQ-How do I control formatting and appearance?," Silmaril Consultants, [Online]. Available: <http://xml.silmaril.ie/style.html>.
- [11] "Associating Style Sheets with XML documents 1.0 (Second Edition)," W3C, [Online]. Available: <http://www.w3.org/TR/xml-stylesheet/>.
- [12] T. Segaran, C. Evans and J. Taylor, *Programming the Semantic Web*, O'Reilly, 2009.
- [13] "Semantic Web Activity," W3C, [Online]. Available: <http://www.w3.org/2001/sw/>.
- [14] A. Gerber, A. van der Merwe and A. Barnard, "A Functional Semantic Web Architecture," [Online]. Available: <http://ksg.meraka.org.za/~agerber/Paper152.pdf>.
- [15] S. Rudi and S. Grimm, *Semantic web services: Concepts, Technologies and Applications*, Springer, 2007.
- [16] J. Davies, D. Fensel and F. van Harmelen, *Towards The Semantic Web: Ontology-driven Knowledge Management*, Wiley, 2003.
- [17] "Ontology (information science)," Wikipedia, [Online]. Available: [http://en.wikipedia.org/wiki/Ontology_\(information_science\)](http://en.wikipedia.org/wiki/Ontology_(information_science)).
- [18] P. Gassner, "Introduction to Linked Open Data for Visualization Creators," [Online]. Available: http://datavisualization.ch/opinions/introduction-to-linked-data/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed:+Datavisualization+Datavisualization.ch.
- [19] R. Cyganiak and A. Jentzsch, "The Linking Open Data cloud diagram," [Online]. Available: <http://richard.cyganiak.de/2007/10/lod/>.
- [20] M. Hausenblas, "Linked Open Data star scheme by example," [Online]. Available: <http://lab.linkeddata.deri.ie/2010/star-scheme-by-example/>.
- [21] "N-Triples," W3C, [Online]. Available: <http://www.w3.org/TR/rdf-testcases/#ntriples>.
- [22] "Notation3 (N3): A readable RDF syntax," W3C, [Online]. Available:

- <http://www.w3.org/TeamSubmission/n3/>.
- [23] "Turtle - Terse RDF Triple Language," W3C, [Online]. Available: <http://www.w3.org/TeamSubmission/turtle/>.
 - [24] "Introducing JSON," [Online]. Available: <http://www.json.org/>.
 - [25] "RDF and JSON: A Clash of Model and Syntax," [Online]. Available: <http://www.ldodds.com/blog/2010/12/rdf-and-json-a-clash-of-model-and-syntax/>.
 - [26] A. Papantoniou, *Knowledge technologies and the Semantic Web-Lecture notes*, 2010.
 - [27] "Dublin Core Metadata Element Set, Version 1.1," DCMI, [Online]. Available: <http://dublincore.org/documents/dces/>.
 - [28] "Namespace Policy for the Dublin Core Metadata Initiative (DCMI)," DCMI, [Online]. Available: <http://dublincore.org/documents/dcmi-namespace/>.
 - [29] "FOAF Vocabulary Specification 0.98," W3C, [Online]. Available: <http://xmlns.com/foaf/spec/>.
 - [30] "Agreement between DCMI and the FOAF Project," DCMI, [Online]. Available: <http://dublincore.org/documents/dcmi-foaf/>.
 - [31] "The DBpedia Data Set," DBpedia, [Online]. Available: <http://wiki.dbpedia.org/Datasets>.
 - [32] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak and S. Hellmann, "DBpedia - A Crystallization Point for the Web of Data," [Online]. Available: <http://www.wiwi.fu-berlin.de/en/institute/pwo/bizer/research/publications/Bizer-et-al-DBpedia-CrystallizationPoint-JWS-Preprint.pdf>.
 - [33] J. Lehmann, J. Schuppel and S. Auer, "Discovering Unknown Connections-the DBpedia Relationship Finder," [Online]. Available: <http://www.jens-lehmann.org/files/2007/relfinder.pdf>.
 - [34] "Visual Data Web-Visually experiencing the Data Web," DEI-Interactive Systems-, [Online]. Available: <http://www.visualdataweb.org/relfinder/relfinder.php>.
 - [35] "SKOS Core Guide," W3C. [Online]. Available: <http://www.w3.org/TR/2005/WD-swbpskos-core-guide-20050510/>.
 - [36] "SKOS Simple Knowledge Organization System Namespace Document," W3C. [Online]. Available: <http://www.w3.org/2009/08/skos-reference/skos.html>.
 - [37] "SKOS Core Guide," W3C, [Online]. Available: <http://www.w3.org/TR/2005/WD-swbpskos-core-guide-20051102/>.
 - [38] L. Feigenbaum and E. Prud'hommeaux, "SPARQL by Example," Cambridge Semantics, [Online]. Available: [http://www.cambridgesemantics.com/2008/09/sparql-by-example/#\(1\)](http://www.cambridgesemantics.com/2008/09/sparql-by-example/#(1)).
 - [39] D. Beckett, "Learn about SPARQL 1.1," May 2011. [Online]. Available: <http://www.dajobe.org/talks/201105-sparql-11/>.
 - [40] "Geocoding," Wikipedia, [Online]. Available: <http://en.wikipedia.org/wiki/Geocoding>.
 - [41] "Google Maps," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Google_Maps.
 - [42] "About Google Maps," Google, [Online]. Available: <http://support.google.com/maps/bin/answer.py?hl=en&answer=7060>.
 - [43] "Google Latitude," Wikipedia, [Online]. Available:

- http://en.wikipedia.org/wiki/Google_Latitude.
- [44] "Skyhook," Skyhook Wireless, [Online]. Available: <http://www.skyhookwireless.com/>.
 - [45] "Frequently Asked Questions," Skyhook Wireless, [Online]. Available: <http://skyhookwireless.com/howitworks/faq.php>.
 - [46] "Google Badge," Google, [Online]. Available: <https://www.google.com/latitude/b/0/apps>.
 - [47] "PHP," The PHP Group, [Online]. Available: <http://www.php.net>.
 - [48] "PHP," Wikipedia, [Online]. Available: <http://en.wikipedia.org/wiki/PHP>.
 - [49] E. Vander Veer, Javascript for Dummies, 4th ed., Wiley Publishing Inc., 2005.
 - [50] M. Haverbeke, Eloquent JavaScript-A Modern Introduction to Programming, No Starch Press, Inc, 2011.
 - [51] "JavaScript," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/JavaScript#Syntax_and_semantics.
 - [52] "Easier way to use Google Latitude API : Google Badge API," BEST FROM GOOGLE, [Online]. Available: <http://bestfromgoogle.blogspot.com/2011/05/easier-way-to-use-google-latitude-api.html>.
 - [53] "Why use JSON over XML?," JQuery4u, [Online]. Available: <http://www.jquery4u.com/json/json-vs-xml/>.
 - [54] blog, "1 in 5 APIs Say "Bye XML"," ProgrammableWeb, 2012. [Online]. Available: <http://blog.programmableweb.com/2011/05/25/1-in-5-apis-say-bye-xml/>.
 - [55] "The Case For JSON: What Is It and Why Use It?," BorkWeb, [Online]. Available: <http://borkweb.com/story/the-case-for-json-what-is-it-and-why-use-it>.
 - [56] "JSON," Wikipedia, [Online]. Available: <http://en.wikipedia.org/wiki/JSON#XML>.
 - [57] "How to Calculate the Distance Between Latitude Lines," Demand Media, Inc, [Online]. Available: http://www.ehow.com/how_6217130_calculate-distance-between-latitude-lines.html.
 - [58] "Virtuoso Universal Server," OpenLink Software, 2010. [Online]. Available: <http://virtuoso.openlinksw.com/>.
 - [59] "Geo-coordinates," DBpedia, [Online]. Available: <http://wiki.dbpedia.org/Datasets#h18-17>.
 - [60] "W3C Semantic Web Interest Group," W3C, [Online]. Available: <http://www.w3.org/2003/01/geo/>.
 - [61] J. Wright, "SPARQL Query In Code: REST, PHP And JSON [TUTORIAL]," [Online]. Available: <http://johnwright.me/blog/sparql-query-in-code-rest-php-and-json-tutorial/>.
 - [62] S. Chandra, "Google Maps API V3 for ASP.NET," 2011. [Online]. Available: <http://www.codeproject.com/Articles/291499/Google-Maps-API-V3-for-ASP-NET>.
 - [63] "Google Maps JavaScript API V3," Google, [Online]. Available: <http://code.google.com/intl/el/apis/maps/documentation/javascript/reference.html>.
 - [64] "Google Maps API Tutorial," [Online]. Available: <http://econym.org.uk/gmap/>.
 - [65] A. Papantoniou and V. Loumos, "A Framework for Visualizing the Web of Data: Combining DBpedia and open APIs".

- [66] G. Svennerberg , "Google Maps API 3 – The basics," In Usability We Trust, [Online]. Available: <http://www.svennerberg.com/2009/06/google-maps-api-3-the-basics/>.
- [67] "Domains, Websites & everything in between," Go Daddy Operating Company, [Online]. Available: <http://www.godaddy.com/>.
- [68] "Linking Open Government Data," Tetherless World Constellation, [Online]. Available: <http://logd.tw.rpi.edu/>.
- [69] "RIF Overview," W3C, [Online]. Available: <http://www.w3.org/TR/2010/NOTE-rif-overview-20100622/>.