Dynamic Web Development

Lecture 7

The Semantic Web 2

http://www.linkeddatatools.com/semantic-web-basics

http://www.w3.org/2009/Talks/0615-qbe/

The Semantic Web

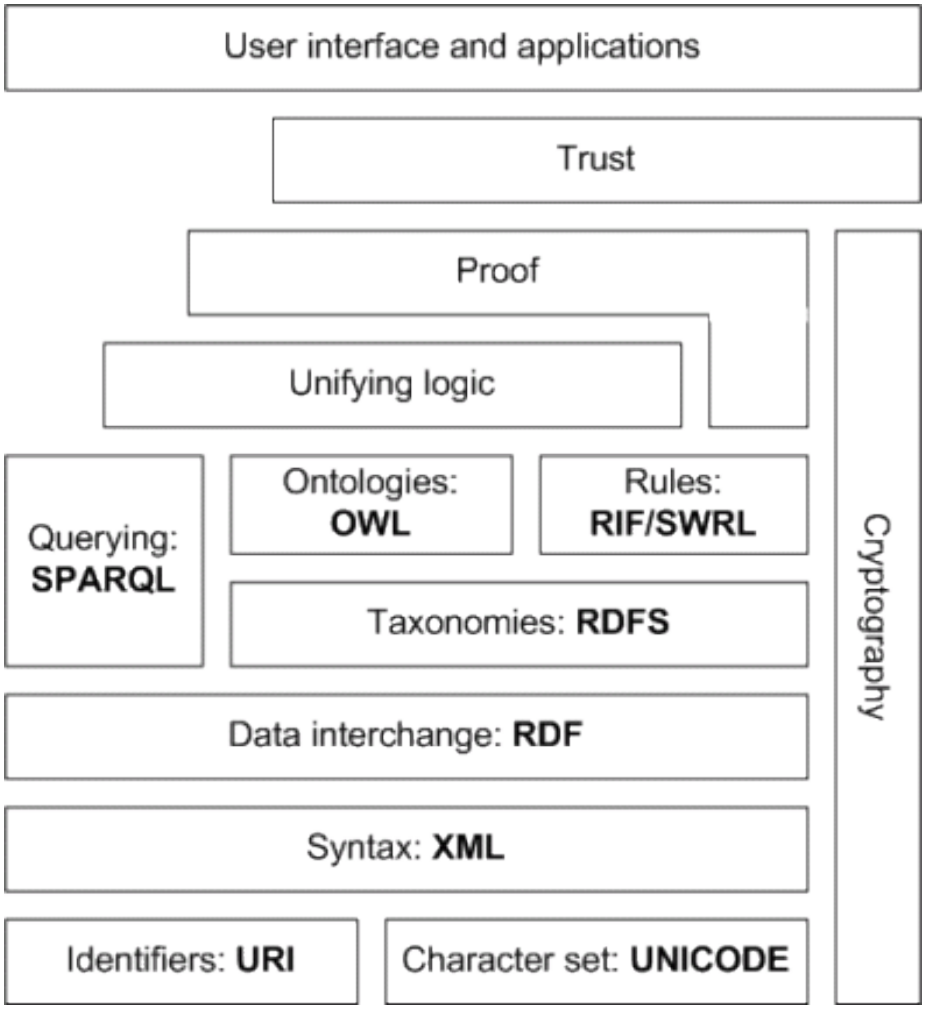
"In addition to the classic “Web of documents” W3C is helping to build a technology stack to support a “Web of data,” the sort of data you find in databases. "

"The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term “Semantic Web” refers to W3C’s vision of the Web of linked data. "

"Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS."

www.w3.org/standards/semanticweb

Semantic Web Diagram



RDF
Resource Description Framework

RDFS
RDF Schema

OWL
Web Ontology Language

RIF
Rule Interchange Format

SPARQL
An RDF query language

http://www.obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

Graph Data

Introducing RDF

Semantic Modelling

- RDFS and OWL

Querying Semantic Data

Other Serialisation Formats

This triple is written in RDF/XML

What does it represent?

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

  <rdf:Description rdf:about="http://example.org/dog1">
     <rdf:type rdf:resource="http://example.org/animal"></rdf:type>
  </rdf:Description>

</rdf:RDF>
```

This was the first standard format for serialising RDF, but there are others:

Turtle	Compact, human-friendly
N-Triples	Easy to process line-based format
JSON-LD	JSON based format
N3	Notation 3 - more powerful version of Turtle

Turtle

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

  <rdf:Description rdf:about="http://example.org/dog1">
     <rdf:type rdf:resource="http://example.org/animal"></rdf:type>
  </rdf:Description>

</rdf:RDF>
```

In Turtle, the above triple would be written as:

```
@PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@PREFIX ex:   <http://example.org/> .

ex:dog1      rdf:type      ex:animal .
```

Much simpler - so why not use it all the time?

Having introduced the advantages of modeling vocabulary and semantics in data models, let's have a look at the actual technology used to add semantics to RDF data models.

RDF data can be encoded with semantic metadata using two languages:

- RDFS
- OWL

RDFS extends RDF, and OWL extends RDFS

RDF Schema Elements

Elements for defining Classes

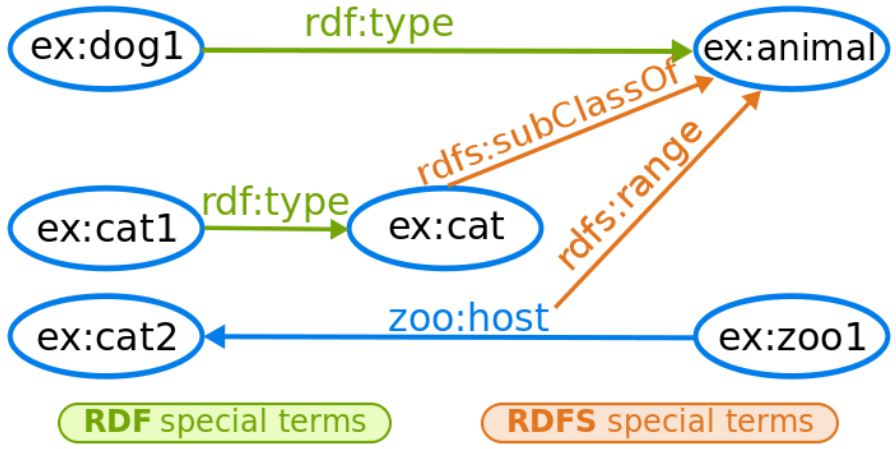
rdfs:Resource
rdfs:Literal
rdfs:Class
rdfs:Datatype
rdfs:Container
rdfs:ContainerMembershipProperty

Elements for defining Properties

rdfs:subClassOf
rdfs:subPropertyOf
rdfs:domain
rdfs:range
rdfs:label
rdfs:comment
rdfs:member
rdfs:seeAlso
rdfs:isDefinedBy

Using Turtle to keep it simple

```
ex:dog1     rdf:type          ex:animal
ex:cat1     rdf:type          ex:cat
zoo:host    rdfs:range        ex:animal
ex:zoo1     zoo:host          ex:cat2
ex:cat      rdfs:subClassOf   ex:animal
```



OWL Elements

For describing Properties
or more formally rdf:Property

owl:allValuesFrom
owl:backwardCompatibleWith
owl:cardinality
owl:complementOf
owl:differentFrom
owl:disjointWith
owl:distinctMembers
owl:equivalentClass
owl:equivalentProperty
owl:hasValue
owl:imports
owl:incompatibleWith
owl:intersectionOf
owl:inverseOf
owl:maxCardinality
owl:minCardinality
owl:oneOf
owl:onProperty
owl:priorVersion
owl:sameAs
owl:someValuesFrom
owl:unionOf
owl:versionInfo

OWL Elements

```
owl:AllDifferent
owl:AnnotationProperty
owl:Class
owl:DataRange
owl:DatatypeProperty
owl:DeprecatedClass
owl:DeprecatedProperty
owl:FunctionalProperty
owl:InverseFunctionalProperty
owl:Nothing
owl:ObjectProperty
owl:Ontology
owl:OntologyProperty
owl:Restriction
owl:SymmetricProperty
owl:Thing
owl:TransitiveProperty
```

For describing Classes

or more formally

rdfs:Class

A First Example

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:owl="http://www.w3.org/2002/07/owl#"
          xmlns:dc="http://purl.org/dc/elements/1.1/">

  <!-- OWL Header Example -->
  <owl:Ontology rdf:about="http://www.linkeddatatools.com/plants">
    <dc:title>The LinkedDataTools.com Example Plant Ontology</dc:title>
    <dc:description>An example ontology written for the LinkedDataTools.com RDFS &
                    OWL introduction tutorial</dc:description>
  </owl:Ontology>

  <!-- OWL Class Definition Example -->
  <owl:Class rdf:about="http://www.linkeddatatools.com/plants#planttype">
    <rdfs:label>The plant type</rdfs:label>
    <rdfs:comment>The class of plant types.</rdfs:comment>
  </owl:Class>

</rdf:RDF>
```

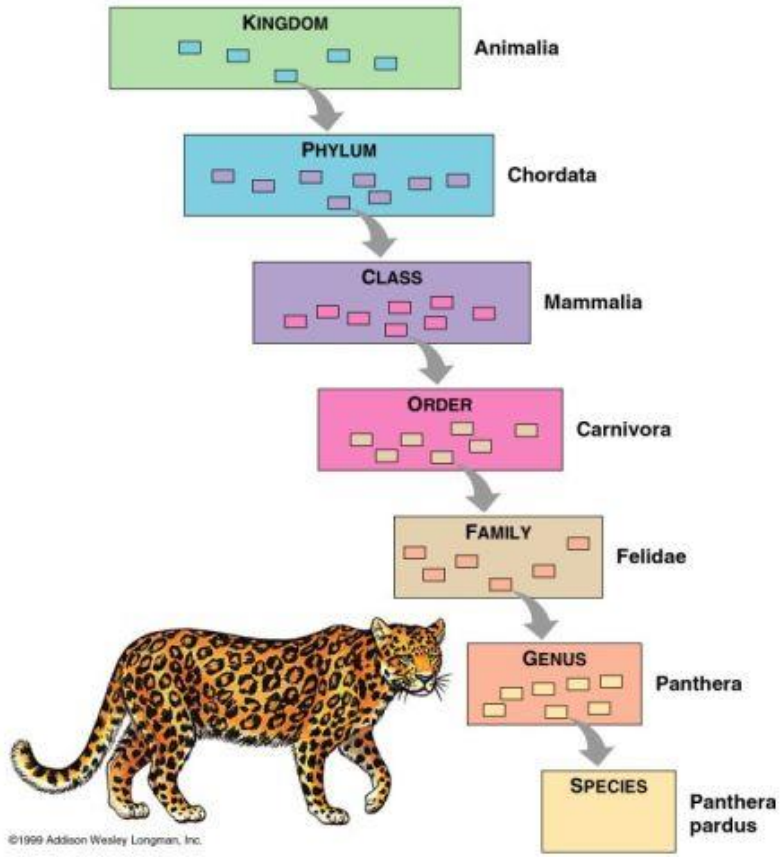
OWL Classes, Subclasses & Individuals

The primary purpose of your ontology is to classify things in terms of semantics, or meaning.

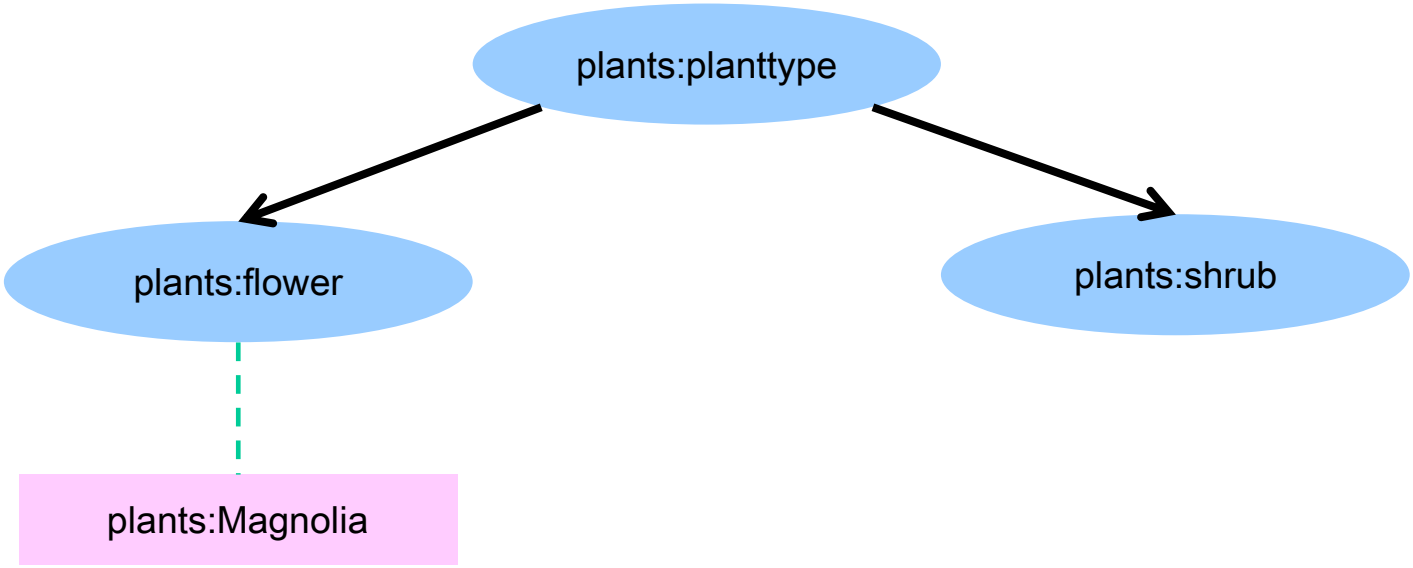
In OWL, this is achieved through the use of *classes* and *subclasses*, instances of which in OWL are called *individuals*.

The individuals that are members of a given OWL class are called its *class extension*.

A *class* in OWL is a classification of individuals into groups which share common characteristics.



Taxonomy - A Hierarchy Of Terms



Why is **Magnolia** in a different shaped box?

It is an individual (instance) of the class **flower**, but not a further subclassification

```
<rdf:RDF   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
           xmlns:owl="http://www.w3.org/2002/07/owl#"
           xmlns:dc="http://purl.org/dc/elements/1.1/"
           xmlns:plants="http://www.linkeddatatools.com/plants#">

  <!-- OWL Header Omitted For Brevity -->

  <!-- OWL Class Definition - Plant Type -->
  <owl:Class rdf:about="http://www.linkeddatatools.com/plants#planttype">

      <rdfs:label>The plant type</rdfs:label>
      <rdfs:comment>The class of all plant types.</rdfs:comment>

  </owl:Class>

  <!-- OWL Subclass Definition - Flower -->
  <owl:Class rdf:about="http://www.linkeddatatools.com/plants#flowers">

      <!-- Flowers is a subclassification of planttype -->
      <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>

      <rdfs:label>Flowering plants</rdfs:label>
      <rdfs:comment>Flowering plants, also known as angiosperms.</rdfs:comment>

  </owl:Class>
```

```
<!-- OWL Subclass Definition - Shrub -->
   <owl:Class rdf:about="http://www.linkeddatatools.com/plants#shrubs">

      <!-- Shrubs is a subclassification of planttype -->
      <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>

       <rdfs:label>Shrubbery</rdfs:label>
       <rdfs:comment>Shrubs, a type of plant which branches from the base.</rdfs:comment>

   </owl:Class>

   <!-- Individual (Instance) Example RDF Statement -->
   <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">

   <!-- Magnolia is a type (instance) of the flowers classification -->
   <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>

  </rdf:Description>

</rdf:RDF>
```

OWL Properties

Individuals in OWL are related by *properties*. There are two types of property in OWL:

Object properties (owl:ObjectProperty) relates individuals (instances) of two OWL classes.

Datatype properties (owl:DatatypeProperty) relates individuals (instances) of OWL classes to literal values.

Let's add

- the name of the class that Magnolia is an instance of - flowers
- a *data type* property which represents the family that Magnolia belongs to - **`Magnoliaceae`**

```
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:owl="http://www.w3.org/2002/07/owl#"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:plants="http://www.linkeddatatools.com/plants#">

  <!-- OWL Header Omitted For Brevity -->

  <!-- OWL Classes Omitted For Brevity -->

  <!-- Where is the property 'family' defined? -->
  <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>

    <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">

          <!-- Magnolia is a type (instance) of the flowers class -->
          <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>

          <!-- The magnolia is part of the 'Magnoliaceae' family -->
          <plants:family>Magnoliaceae</plants:family>

    </rdf:Description>

</rdf:RDF>
```

Important Point

At this point, if you are an object oriented programmer, your mind may well be thinking of programmatic object classes and their associated properties and comparing them to what we've just learned out OWL classes.

Be careful - they're not quite the same.

Note from the example above. The 'family' property was defined independent of any class type, and was assigned to the *instance* of class **flower** (magnolia).

Another instance of the same class may not have this property. So in OWL, note that the properties that instances have, are not described in their *class types*, but their *instances*.

The strong inheritance of OOP classes doesn't work in the same way here.

In this case, you may use the same 'family' property for an instance of a completely different class

W3C Validation

You can investigate the structure of an RDF document by submitting it to the W3C RDF Validator at:

http://www.w3.org/RDF/Validator

It will display a list of the triples and also a graph diagram.

```
<rdf:RDF
  <!-- Namespaces Omitted For Brevity -->
  <!-- OWL Header Omitted For Brevity -->
   <!-- OWL Classes Omitted For Brevity -->

<!-- Define the family property -->
<owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>

<!-- Define the similarlyPopularTo property -->
<owl:ObjectProperty   rdf:about="http://www.linkeddatatools.com/plants#similarlyPopularTo"/>

  <!-- Define the Orchid class instance -->
  <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#orchid">

    <!-- Orchid is an individual (instance) of the flowers class -->
    <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>

    <!-- The orchid is part of the 'Orchidaceae' family -->
    <plants:family>Orchidaceae</plants:family>

    <!-- The orchid is similarly popular to the magnolia -->
    <plants:similarlyPopularTo  rdf:resource="http://www.linkeddatatools.com/plants#magnolia"/>

  </rdf:Description>

<!-- Define the Magnolia class instance -->
<rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">

  <!-- Magnolia is an individual (instance) of the flowers class -->
  <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>

  <!-- The magnolia is part of the 'Magnoliaceae' family -->
  <plants:family>Magnoliaceae</plants:family>

  <!-- The magnolia is similarly popular to the orchid -->
  <plants:similarlyPopularTo  rdf:resource="http://www.linkeddatatools.com/plants#orchid"/>

</rdf:Description>

</rdf:RDF>
```

Output from W3C Validator

Number	Subject	Predicate	Object
1	http://www.linkeddatatools.com/plants#planttype	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
2	http://www.linkeddatatools.com/plants#planttype	http://www.w3.org/2000/01/rdf-schema#label	"The plant type"
3	http://www.linkeddatatools.com/plants#planttype	http://www.w3.org/2000/01/rdf-schema#comment	"The class of all plant types."
4	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
5	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/plants#planttype
6	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/2000/01/rdf-schema#label	"Flowering plants"
7	http://www.linkeddatatools.com/plants#flowers	http://www.w3.org/2000/01/rdf-schema#comment	"Flowering plants, also known as angiosperms."
8	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
9	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/2000/01/rdf-schema#subClassOf	http://www.linkeddatatools.com/plants#planttype
10	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/2000/01/rdf-schema#label	"Shrubbery"
11	http://www.linkeddatatools.com/plants#shrubs	http://www.w3.org/2000/01/rdf-schema#comment	"Shrubs, a type of plant which branches from the base."
12	http://www.linkeddatatools.com/plants#magnolia	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.linkeddatatools.com/plants#flowers

Output from the W3C Validator

Inferencing

RDF datastores can return results that are not directly stored as data, but which are implied by the metadata. Say you had included metadata which represented:

- Sean Connery is the star of Goldfinger
- Goldfinger is one of the Bond Films
- A star is the main actor of a film
- Actors are often a member of Equity
- All Bond films only use Equity approved actors.

There is no triple which encodes the information that Sean Connery is a member of Equity.

But, a query which was designed to list all members of Equity would show Sean Connery.

A relational database cannot do that - unless you construct a horribly complicated query. Why would the query need to be complicated?

An RDF triplestore does it automatically.

Graph Data

Introducing RDF

Semantic Modelling

RDFS and OWL

- Querying Semantic Data

Publically Available Datasets

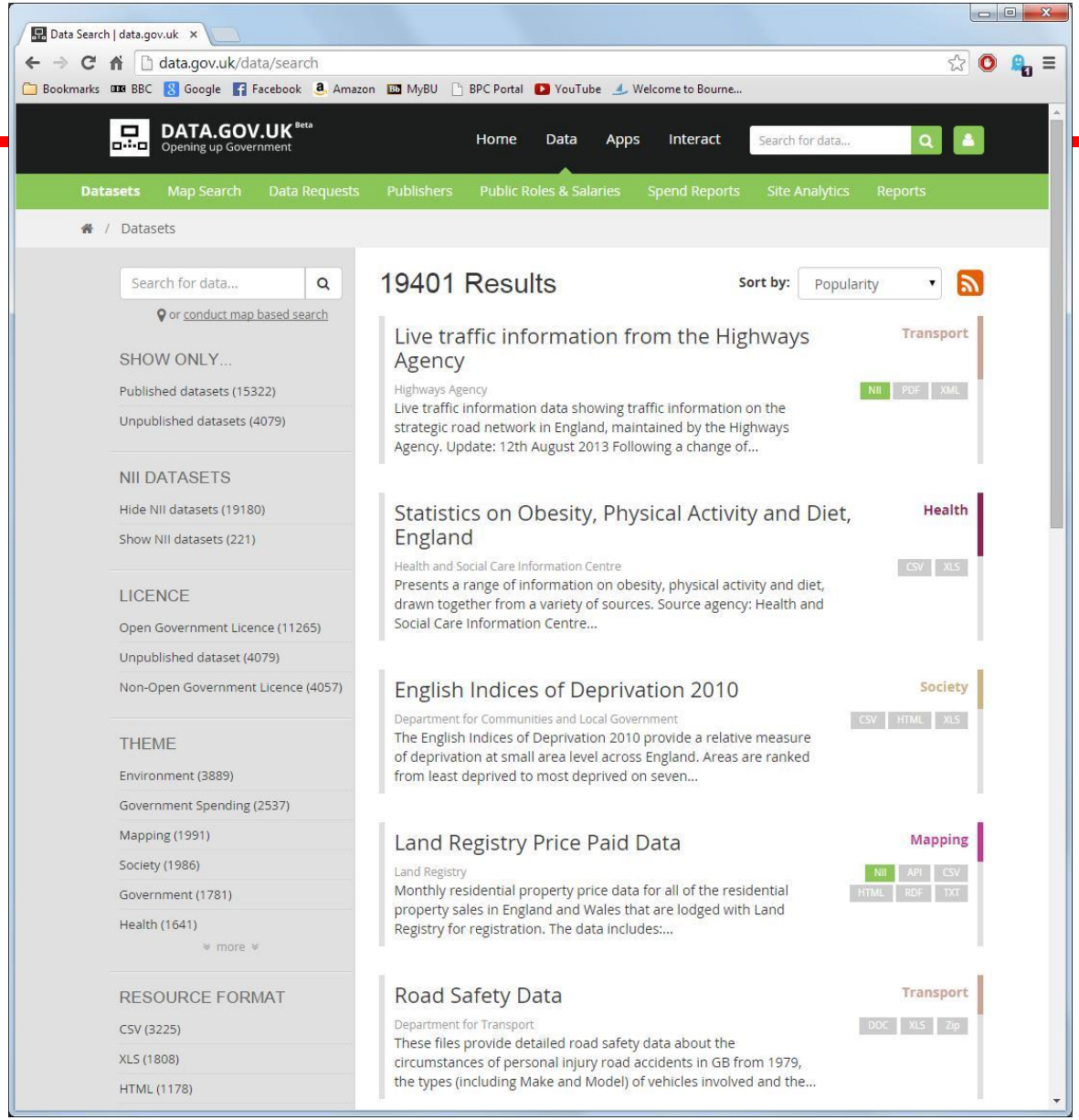
The Government makes a number of datasets available in varying formats:

http://data.gov.uk

The datasets are listed here:

http://data.gov.uk/data/search

There are many other organisations that are making their data publically available in these new formats.


Data Search | data.gov.uk
data.gov.uk/data/search
Bookmarks BBC Google Facebook Amazon MyBU BPC Portal YouTube Welcome to Bourne...
DATA.GOV.UK Beta
Opening up Government
Home Data Apps Interact
Search for data...
Datasets Map Search Data Requests Publishers Public Roles & Salaries Spend Reports Site Analytics Reports
/ Datasets
Search for data...
or conduct map based search
SHOW ONLY...
Published datasets (15322)
Unpublished datasets (4079)
NII DATASETS
Hide NII datasets (19180)
Show NII datasets (221)
LICENCE
Open Government Licence (11265)
Unpublished dataset (4079)
Non-Open Government Licence (4057)
THEME
Environment (3889)
Government Spending (2537)
Mapping (1991)
Society (1986)
Government (1781)
Health (1641)
more
RESOURCE FORMAT
CSV (3225)
XLS (1808)
HTML (1178)
19401 Results
Sort by: Popularity
Live traffic information from the Highways Agency
Transport
Highways Agency
NII PDF XML
Live traffic information data showing traffic information on the strategic road network in England, maintained by the Highways Agency. Update: 12th August 2013 Following a change of...
Statistics on Obesity, Physical Activity and Diet, England
Health
Health and Social Care Information Centre
CSV XLS
Presents a range of information on obesity, physical activity and diet, drawn together from a variety of sources. Source agency: Health and Social Care Information Centre...
English Indices of Deprivation 2010
Society
Department for Communities and Local Government
CSV HTML XLS
The English Indices of Deprivation 2010 provide a relative measure of deprivation at small area level across England. Areas are ranked from least deprived to most deprived on seven...
Land Registry Price Paid Data
Mapping
Land Registry
NII API CSV HTML RDF TXT
Monthly residential property price data for all of the residential property sales in England and Wales that are lodged with Land Registry for registration. The data includes:...
Road Safety Data
Transport
Department for Transport
DOC XLS Zip
These files provide detailed road safety data about the circumstances of personal injury road accidents in GB from 1979, the types (including Make and Model) of vehicles involved and the...

Querying Semantic Data

RDF data stores can also be queried using their own query language - SPARQL (SPARQL Protocol and RDF Query Language, pronounced "sparkle").

SPARQL is a W3C standard and is currently at version 1.1.

Just to show you what SPARQL looks like, here is a quick example:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>

SELECT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .
    FILTER (?population > 15000000) .
}
```

This query will return the names of all landlocked countries with a population greater than 15,000,000.

What dataset are we running it against?

DBPedia

Copy and paste the above query into this webpage

http://dbpedia.org/snorql/

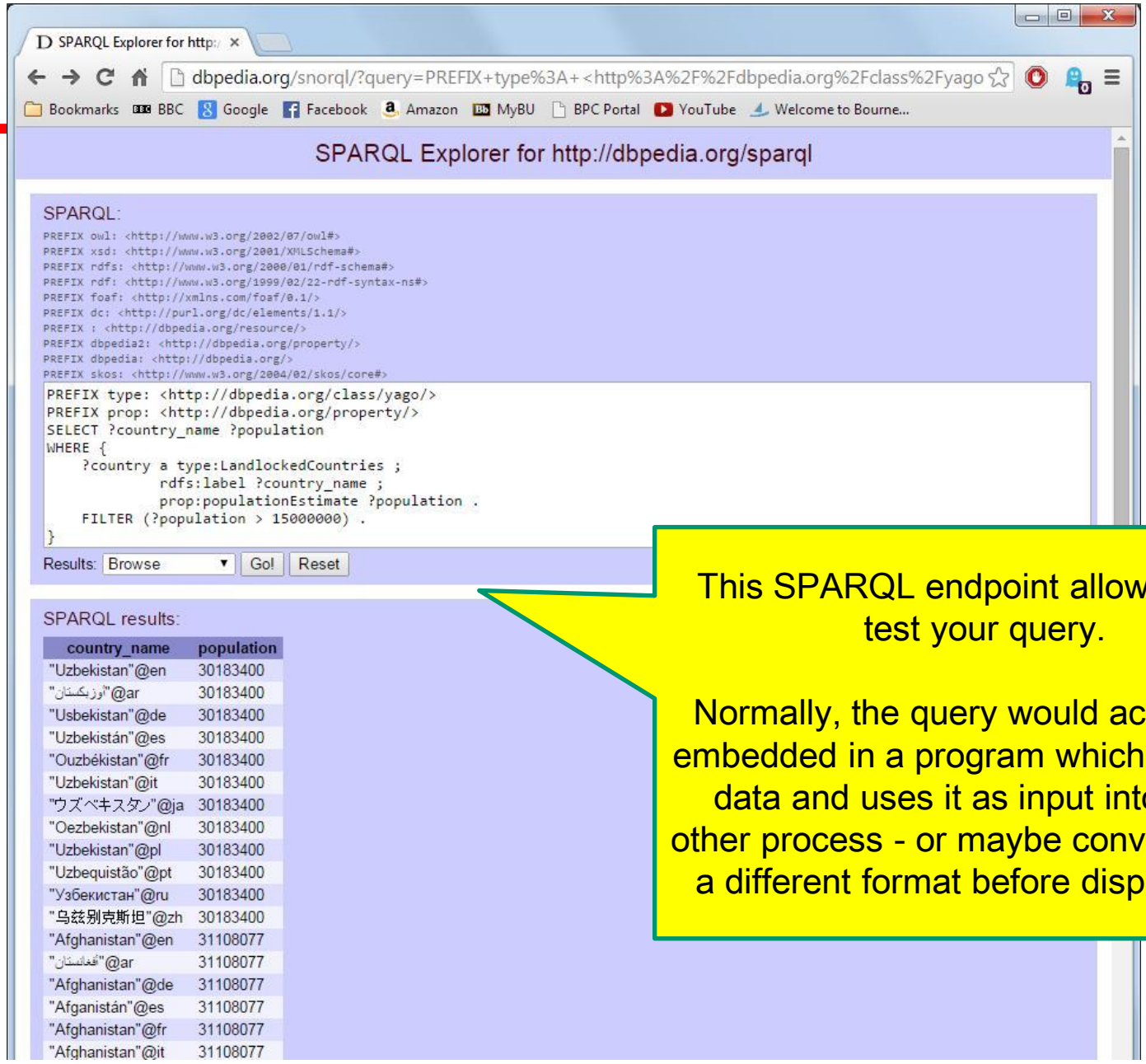
and the see the results.

This URL is called a **SPARQL endpoint** in the semantic web world - and is the way in which the data on the semantic web is published to the outside world in a query enabled form.

DBPedia is an RDF version of information from Wikipedia.

DBPedia contains data derived from Wikipedia's infoboxes, category hierarchy, article abstracts, and various external links.

DBpedia contains over 100 million triples.



This SPARQL endpoint allows you to test your query.

Normally, the query would actually be embedded in a program which takes the data and uses it as input into some other process - or maybe converts it into a different format before displaying it.

SPARQL General Structure

A SPARQL query comprises, in order:

Prefix declarations, for abbreviating URIs
Dataset definition, stating what RDF graph(s) are being queried
A result clause, identifying what information to return from the query
The query pattern, specifying what to query for in the underlying dataset
Query modifiers, slicing, ordering, and otherwise rearranging query results

```
         # prefix declarations
PREFIX foo: <http://example.com/resources/>
...
         # dataset definition
FROM ...
         # result clause
SELECT ...
         # query pattern
WHERE {
    ...
}
         # query modifiers
ORDER BY ...
```

What is this query doing?

```
PREFIX abc: <http://example.com/exampleOntology#>

SELECT ?capital ?country
WHERE
{
      ?x abc:cityname ?capital .
      ?x abc:isCapitalOf ?y .
      ?y abc:countryname ?country .
      ?y abc:isInContinent abc:Africa .
 }
ORDER BY ?capital
```

What About CREATE, INSERT, UPDATE?

You may have noticed that whilst we have covered the basics of the **SELECT-WHERE** form of SPARQL to obtain subsets of data from a triple data store, we haven't yet covered the **CREATE**, **INSERT** or **UPDATE** methods. There's a good reason for this; at the moment, they are not implemented.

If you think about it, at some level it makes sense. Organisations publishing data in a queryable, public SPARQL endpoint will probably in most circumstances not just want anyone writing and making changes to their (valuable) data.

But also, at some level, it does seem like it's a missing requirement, particularly if you're using triple data stores locally rather than publicly across the web and want to make changes using a query language (like you would with a SQL database). How could this be done?

At the moment, new specifications such as SPARUL (SPARQL/Update) and SPARQL+ are being developed to address this problem, however a solid contender to fill this gap in functionality is yet to arise.

A quick search on 'Using SPARQL' produced this

http://blog.swirrl.com/articles/sparql-example-find-data-for-postcode/

http://data-gov.tw.rpi.edu/wiki/How_to_use_SPARQL

http://answers.semanticweb.com/questions/26621/using-a-sparql-query-to-get-data-from-uk-government-datasets

http://stackoverflow.com/questions/16608265/querying-open-data-communities-data-with-sparql

http://www.linkeddatatools.com/querying-semantic-data

https://jena.apache.org/tutorials/sparql.html

http://www.cambridgesemantics.com/semantic-university/sparql-by-example

http://www.w3.org/2004/Talks/17Dec-sparql/