

# Text Encoding and Semantic Representation

---

Semantic Web

marilena.daquino2@unibo.it | [https://github.com/marilenadaquino/tesr\\_dhdk](https://github.com/marilenadaquino/tesr_dhdk)

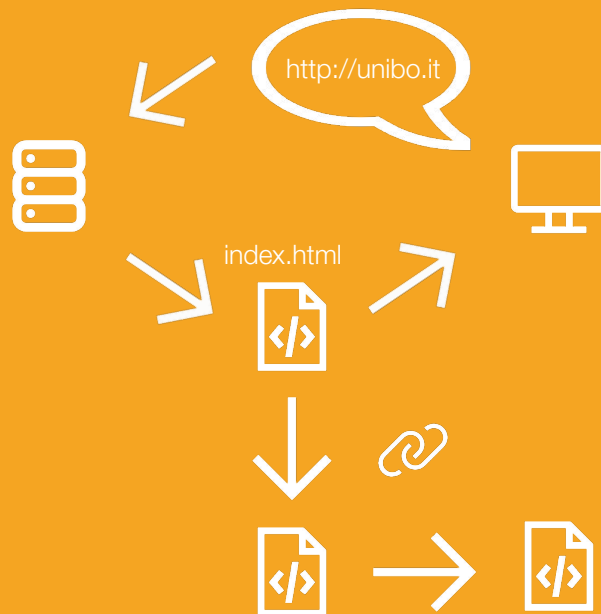


# Semantic Web

## Motivation

The current web is based on interlinked **HTML** (HyperText Markup Language) documents, served on the web via the **HTTP** (HyperText Transfer Protocol) protocol and retrievable via their **URL** (Uniform Resource Locator).

Such documents include a wealth of **unstructured** information that the **human reader** is usually required to parse manually to understand its content and the links to other documents.



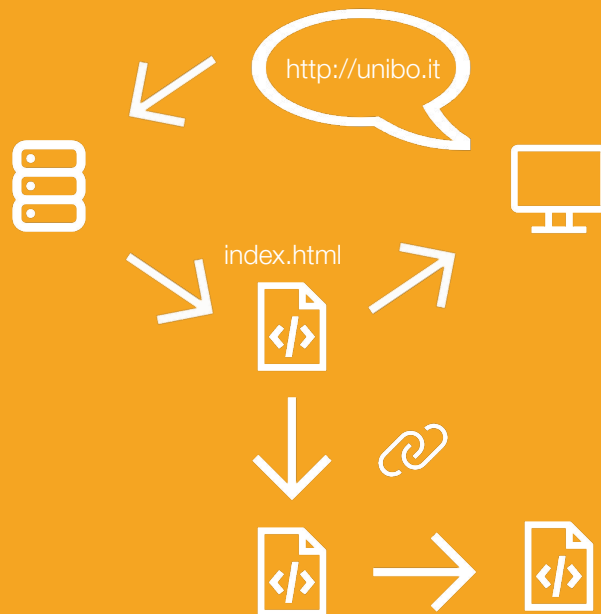
# Semantic Web

---

## Motivation

While **structured** databases exist, these are isolated from similar data and the relations between data points are not understandable by machines.

The idea of Semantic Web is to make information available on the web **understandable by machines**, so as to facilitate every-day information retrieval tasks (i.e. search) and make results more accurate.

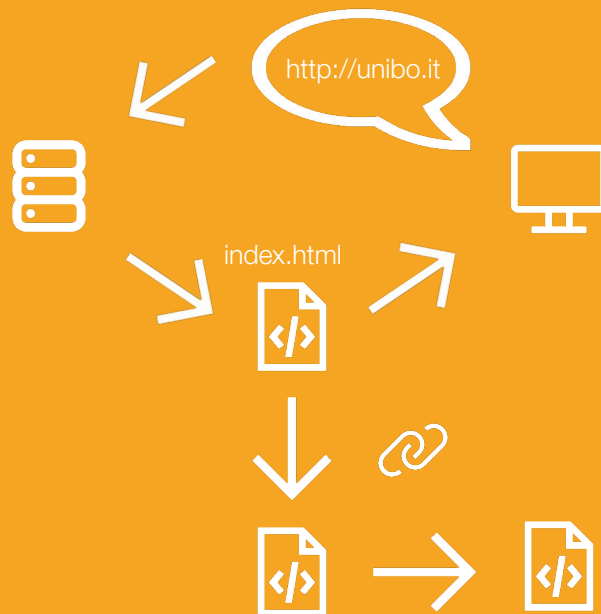


# Semantic Web

---

## Objectives

1. Make structured and semi-structured data available on the web according to standard **formats** (e.g. XML) to facilitate exchange and mashup
2. Make available not only data but also metadata about **relations** between data available on the web
3. Describe the **semantics** associated to data and relations with a formalism that is understandable by machines



# Semantic Web

---

## Main technologies



### Graphs

Data are organised in nodes (data points) and edges (relations).



### Web identifiers

Both data points and their relations are identified with URIs (Uniform Resource Identifiers)



### Ontologies

Vocabularies with definitions of types and relations between data.

# Semantic Web

---

## Main technologies



### Graphs

RDF (Resource Description Framework) is the formalism used to represent graphs in a machine-readable format.



### Web identifiers

URIs are used to identify nodes and edges of the graph. This mechanism is embedded in RDF



### Ontologies

RDF Schema and OWL are languages to formally define types and relations.

# Semantic Web

---

## URIs

URIs, similarly to URLs (locations) and URNs (names), identify resources.

However, **URLs** are meant to provide the address of the digital document without retaining any information about its content.

**URNs** instead, are identifiers associated to the content of a resource (e.g. the ISBN of a book represents a specific edition of that content), but do not tell us where to find the resource.

**URIs** both identify conceptual or real-world entities (e.g. people, places) and tell us where to find data about that concept. Notably the data associated to the entity is provided according to the RDF model.



# Semantic Web

---

## URIs vs URLs

Notice that the URI identifying a concept **does not identify a HTML page** describing the entity.

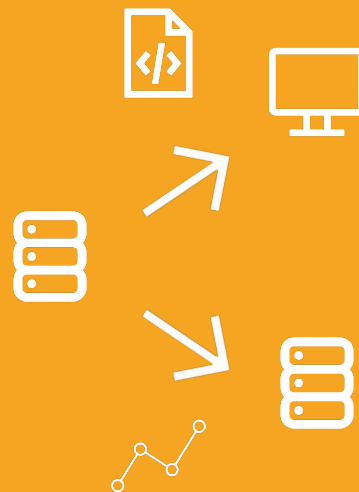
- If the location of a HTML file describing the entity changes, the URI does not change.
- Many HTML pages can describe the same entity identified by the same URI.
- A HTML page can include information about many entities, hence being linked to many URIs.

# Semantic Web

---

## Content negotiation

However when associating a URI to a concept, a mechanism called **content negotiation** is in place. That is, according to who requests the URI, data in a different format is returned, such as a web page, a XML dump of data, or other formats.



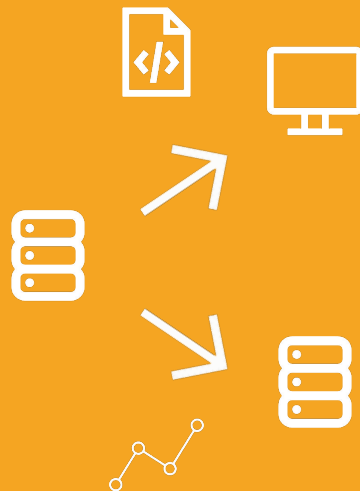
# Semantic Web

---

## Content negotiation

E.g. [https://dbpedia.org/resource/Robert\\_Capa](https://dbpedia.org/resource/Robert_Capa) returns a **web page** (built on top of the graph data) if requested by a browser (i.e. a user)  
Notice it redirects to [https://dbpedia.org/page/Robert\\_Capa](https://dbpedia.org/page/Robert_Capa)

The same URI returns the **RDF** data if requested by a machine. For instance you can request the XML version of data via command-line, which redirects and downloads data available at:  
[https://dbpedia.org/data/Robert\\_Capa.xml](https://dbpedia.org/data/Robert_Capa.xml)



# Exercise

---

## Request RDF data via URI


Open the terminal and check if you have the command **curl**. Type:

- (Windows) **curl -V**
- (Mac) **curl -V**

Or install: <https://curl.se/download.html>

Or download [RDF/XML](#) (also on our repo)

*When you are sure you have curl installed, type and press enter*

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains a single line of text: `1 curl -L -H "Accept: application/rdf+xml;" http://dbpedia.org/resource/Robert_Capa`

```
1 curl -L -H "Accept: application/rdf+xml;" http://dbpedia.org/resource/Robert_Capa
```

# Exercise

## Explore

It's an XML file

- Prologue
- Root element
- Namespaces
- 2 <rdf:Description>
- @rdf:about
- @rdf:resource

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5   xmlns:foaf="http://xmlns.com/foaf/0.1/"
6   xmlns:dbp="http://dbpedia.org/property/"
7   xmlns:dbo="http://dbpedia.org/ontology/"
8   xmlns:dcterms="http://purl.org/dc/terms/"
9   xmlns:owl="http://www.w3.org/2002/07/owl#"
10  xmlns:gold="http://purl.org/linguistics/gold/"
11  xmlns:schema="http://schema.org/"
12  xmlns:prov="http://www.w3.org/ns/prov#" >
13  <rdf:Description rdf:about="http://dbpedia.org/resource/Camera_(magazine)">
14    <dbo:wikiPageWikiLink rdf:resource="http://dbpedia.org/resource/Robert_Capa" />
15  </rdf:Description>
16  ...
17  <rdf:Description rdf:about="http://dbpedia.org/resource/Robert_Capa">
18    <rdf:type rdf:resource="http://dbpedia.org/ontology/Person"/>
19    <rdfs:label xml:lang="ga">Robert Capa</rdfs:label>
20    <foaf:depiction rdf:resource="http://commons.wikimedia.org/wiki/Special:FilePath/Capa-Haus.jpg"/>
21    <dbo:birthPlace rdf:resource="http://dbpedia.org/resource/Budapest" />
22    <owl:sameAs
      rdf:resource="http://api.nytimes.com/svc/semantic/v2/concept/name/nytd_per/Capa,%20Robert" />
23  </rdf:Description>
24 </rdf:RDF>
```

# Exercise

---

## Explore

- @xml:lang and element value
- <owl:sameAs>

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5   xmlns:foaf="http://xmlns.com/foaf/0.1/"
6   xmlns:dbp="http://dbpedia.org/property/"
7   xmlns:dbo="http://dbpedia.org/ontology/"
8   xmlns:dcterms="http://purl.org/dc/terms/"
9   xmlns:owl="http://www.w3.org/2002/07/owl#"
10  xmlns:gold="http://purl.org/linguistics/gold/"
11  xmlns:schema="http://schema.org/"
12  xmlns:prov="http://www.w3.org/ns/prov#" >
13  <rdf:Description rdf:about="http://dbpedia.org/resource/Camera_(magazine)">
14    <dbo:wikiPageWikiLink rdf:resource="http://dbpedia.org/resource/Robert_Capa" />
15  </rdf:Description>
16  ...
17  <rdf:Description rdf:about="http://dbpedia.org/resource/Robert_Capa">
18    <rdf:type rdf:resource="http://dbpedia.org/ontology/Person"/>
19    <rdfs:label xml:lang="ga">Robert Capa</rdfs:label>
20    <foaf:depiction rdf:resource="http://commons.wikimedia.org/wiki/Special:FilePath/Capa-Haus.jpg"/>
21    <dbo:birthPlace rdf:resource="http://dbpedia.org/resource/Budapest" />
22    <owl:sameAs
      rdf:resource="http://api.nytimes.com/svc/semantic/v2/concept/name/nytd_per/Capa,%20Robert" />
23  </rdf:Description>
24 </rdf:RDF>
```

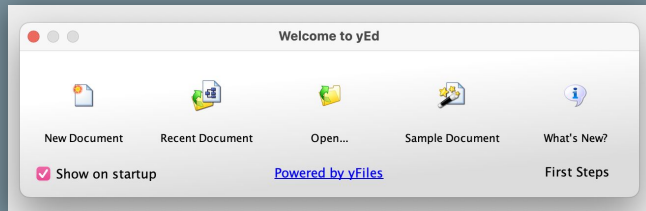
# Exercise

---

## Draw

Let's draw a network graph where our nodes are named with the values of `@rdf:about` and `@rdf:resource` and the edges are named with the tag name of elements children of `rdf:Description`.

- Download [yEd editor](#)
- Create a new document

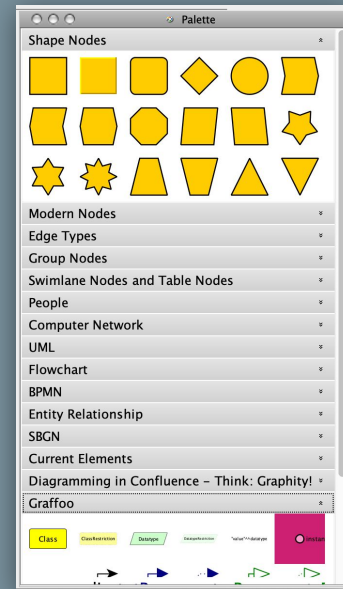


# Exercise

---

## Draw

- Select from the right column the graphic library Graffoo
- Add some individuals (pink circles): Robert Capa, Budapest, magazine...  
Use the value of **@rdf:about** and **@rdf:resource** to name the circles
- Add relations (blue) edges between circles and name them as the element name where **@rdf:resource** appear
- Add a string (green block): use the value of rdfs:label element
- Add a class (yellow block): call it with the value of **rdf:type/@rdf:resource**





# Exercise

---

## Why drawing a graph?

We have studied XML and we have seen that it can be interpreted as a tree, i.e. a hierarchical structure of parent/children elements, which can be seen also as a uni-directed graph. RDF allows you to create **bidirected graphs** (there is not just one hierarchy).

But if XML is our syntax, what is RDF?

# Semantic Web

---

## RDF

Resource Description Framework (RDF) is a model that allows you to **represent knowledge** in the form of graphs. The pillars of a graph are statements, called **triples**, made of <subject> <predicate> <object>

NB. In the RDF/XML file we have seen <rdf:Description> identifies a subject (@rdf:about includes its URI), its children elements are the predicates, and @rdf:resource (or the value of the child element) are the objects



# Semantic Web

---

## RDF + URI

The basic structure `<subject> <predicate> <object>` is composed of **2-3 URIs**, each representing a building block of the triple.

While `@rdf:about` and `@rdf:resource` already mention the URI, it's worth noting that also the predicate (i.e. the relation) can be represented by a URI. In fact, the URI of predicates is made of the namespace URI + the tag name

`<http://dbpedia.org/resource/Robert_Capa> <http://dbpedia.org/resource/Budapest>`



# Semantic Web

---

## RDF + ns

To make the graph and the data more readable,  
full URIs can indeed be replaced by the ns prefix  
+ name

dbr: <http://dbpedia.org/resource/>

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

<dbr:Robert\_Capa>

<dbr:Budapest>



<dbo:birthPlace>

# Semantic Web

---

## RDF Literals

Sometimes the object is not a resource, i.e. a URI, but a string, also called **Literal**.

Several data types exist to characterise the string. Such types are borrowed by XML Schema (xsd) data types, and include integers, dates, etc.

<dbr:Robert\_Capa>

"Robert Capa"



<rdfs:label>

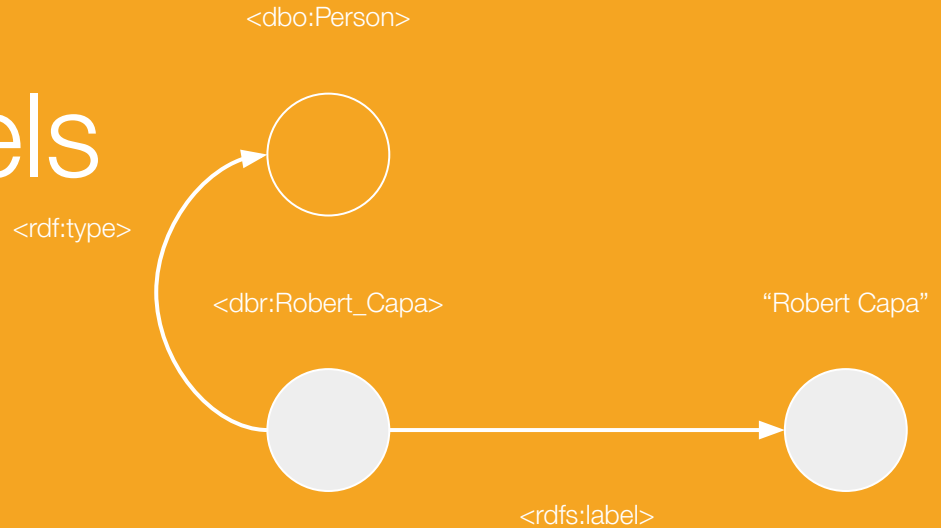
# Semantic Web

---

## Types and labels

A special predicate is **rdf:type**, which allows you to specify one or more types categorizing entities. Such types are called **classes**.

Another special predicate is **rdfs:label**, which associates a human-readable string to URIs.



# Semantic Web

## RDF syntaxes

As aforementioned, rdf is not a syntax. Instead, several syntaxes can be used to represent RDF data, as long as the triplet format is respected.

After **XML**, the simplest notation is **N-Triples**, where s,p, and o URIs are written in full and triples are separated by full stop.

*data.xml*

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <rdf:RDF
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5   xmlns:foaf="http://xmlns.com/foaf/0.1/"
6   xmlns:dbp="http://dbpedia.org/property/"
7   xmlns:dbo="http://dbpedia.org/ontology/"
8   xmlns:dcterms="http://purl.org/dc/terms/"
9   xmlns:owl="http://www.w3.org/2002/07/owl#"
10  xmlns:gold="http://purl.org/linguistics/gold/"
11  xmlns:schema="http://schema.org/"
12  xmlns:prov="http://www.w3.org/ns/prov#" >
13  <rdf:Description rdf:about="http://dbpedia.org/resource/Robert_Capa">
14    <rdf:type rdf:resource="http://dbpedia.org/ontology/Person"/>
15    <rdfs:label xml:lang="ga">Robert Capa</rdfs:label>
16    <dbo:birthPlace rdf:resource="http://dbpedia.org/resource/Budapest" />
17  </rdf:Description>
18 </rdf:RDF>
```

*data.ntriples*

```
1 <http://dbpedia.org/resource/Robert_Capa> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://dbpedia.org/ontology/Person> .
2 <http://dbpedia.org/resource/Robert_Capa> <http://www.w3.org/2000/01/rdf-schema#label> "Robert Capa"@ga .
3 <http://dbpedia.org/resource/Robert_Capa> <http://dbpedia.org/ontology/birthPlace>
  <http://dbpedia.org/resource/Budapest> .
```

# Semantic Web

## RDF syntaxes

A compact version of N-triples is **Turtle**, where full URIs are replaced by their namespaces prefixes and the last part of the URI.

A more recent and convenient syntax is **JSON-LD**. Based on JSON, a format for exchanging data on the web that is “replacing” XML in plenty of contexts, it can be extended to represent triples.

*data.ttl*

```
1 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2 @prefix dbr: <http://dbpedia.org/resource/> .
3 @prefix dbo: <http://dbpedia.org/ontology/> .
4
5 dbr:Robert_Capa a dbo:Person ;
6   rdfs:label "Robert Capa"@ga ;
7   dbo:birthPlace dbr:Budapest .
```

*data.json*

```
1 [
2   {"@id": "http://dbpedia.org/ontology/Person"},
3   {"@id": "http://dbpedia.org/resource/Budapest"},
4   {"@id": "http://dbpedia.org/resource/Robert_Capa",
5     "@type": ["http://dbpedia.org/ontology/Person"],
6     "http://www.w3.org/2000/01/rdf-schema#label": [
7       {"@value": "Robert Capa",
8         "@language": "ga"}
9     ],
10    "http://dbpedia.org/ontology/birthPlace" [
11      {"@id": "http://dbpedia.org/resource/Budapest"}
12    ]
13 ]
14 ]
15 ]
```

*Doesn't it look like a list of dictionaries?*



# Semantic Web

---

## RDF formats

While RDF is not a syntax, .rdf can be used as a **format**. This format can be used regardless of the syntax used in the document, and will be appropriately interpreted by a parser.

Rdf data can be converted losslessly from format to format, e.g. using [converters](#). Similarly, RDF visualisers exist and accept any of the standard syntaxes, e.g. [RDF Grapher](#), [isSemantic](#)

# Exercise

---

## Write your first RDF file

Write as many triples as you can figure from this text in natural language. Write them in turtle and use 2 namespaces of your invention (one for individuals and one for predicates and types).

*Robert Capa is a Hungarian photographer. He was born in 1913.  
His real name is Endre Ernő Friedmann. He became famous for his  
photograph called "Death of a loyalist soldier", which was taken in  
Spain in 1936. It depicts a soldier dying in the Spanish civil war.*

# Semantic Web

---

## Ontologies

Similar to XML schemas, predicates and classes are defined in external vocabularies called **ontologies**, which include terms, the hierarchy, their usage and restrictions (domain and range).

Ontologies are documents (external to data) available online for reuse. Their role is to facilitate **semantic interoperability**, that is, the more people describe their data using the same ontologies, the easier will be to seamlessly integrate different data sources. Content-wise, an ontology describes concepts and properties that characterise a certain **domain** (e.g. Arts, libraries) or a certain task (e.g. provenance). As we have seen, multiple ontologies (namespaces) can be used at the same time to describe data.

# Semantic Web

---

## Ontologies

The definitions and the constraints are written with a dedicated language, called **Web ontology language (OWL)**. Notably an ontology is written using the RDF formalism (subj +pred+obj).

Like RDF, OWL is a formalism to represent knowledge, and several syntaxes can be used to describe an ontology.

*Ontology.rdf or ontology.owl*

```
1 <http://dbpedia.org/ontology/Person>
2 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
3 <http://www.w3.org/2002/07/owl#Class> .
4
5 <http://dbpedia.org/ontology/Person>
6 <http://www.w3.org/2000/01/rdf-schema#label> "Person" .
7
8 <http://dbpedia.org/ontology/Person>
9 <http://www.w3.org/2000/01/rdf-schema#comment> "A human being" .
10
11 <http://example.org/ontology/hasSpouse>
12 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
13 <http://www.w3.org/2002/07/owl#ObjectProperty> .
14
15 <http://example.org/ontology/hasSpouse>
16 <http://www.w3.org/2000/01/rdf-schema#label> "has spouse".
```

# Semantic Web

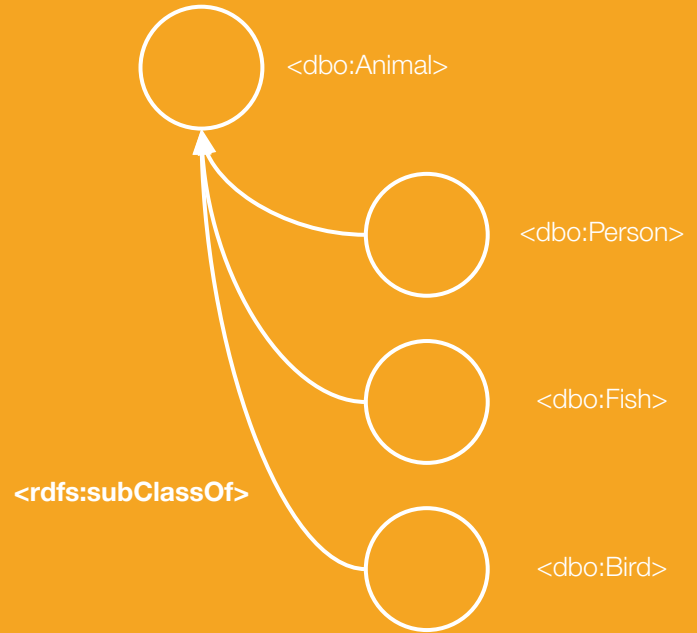
---

## Classes

Are categories describing identified by URIs.

In ontologies it is possible to formalise a hierarchy of concepts via the property **rdfs:subClassOf**.

A basic principle is inheritance, or **subsumption**: an entity that belongs to a subclass also belong to the superclass.



# Semantic Web

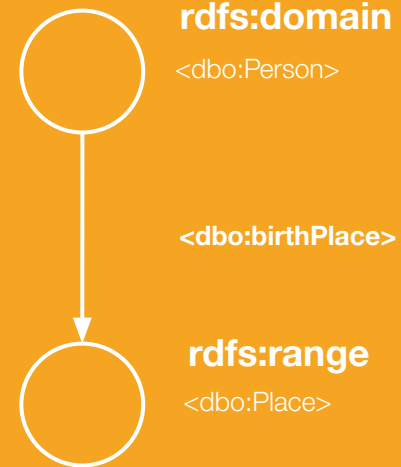
---

## Properties

They represent predicates, that is, relations between entities that belong to classes. The **class** of subjects is called **domain of the property**, and the class of the object is called **range**.

There are two types of properties:

- **Object properties:** connect entities identified with a URI
- **Data properties:** connect a URI to a Literal



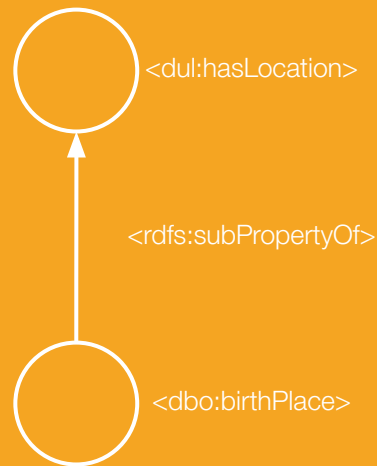
# Semantic Web

---

## Properties

Also properties can be part of a hierarchy, represented via the predicate **rdfs:subPropertyOf**.

Notice in the example that the super property has a different namespace. Indeed, terms belonging to different ontologies can be integrated in the same ontology



# Semantic Web

---

## Ontology alignment

Despite ideally ontologies should be complementary (content-wise), they may present overlap of terms. When the semantics is similar but not a full match, **rdfs:subClassOf/subPropertyOf** are used between terms. When there is a full overlap, special predicates are provided by the OWL language, namely:

- **owl:equivalentClass** (between classes)
- **owl:equivalentProperty** (between properties)
- **owl:sameAs** (between individuals). Notably this predicate is found mostly in datasets rather than in ontologies



# Exercise | Homework

## Protégé

Protégé is an open-source GUI (online and desktop) to develop ontologies. It's easy to install and use.

Let's use protégé to develop an ontology to be used to describe the data in the prior exercise. Include alignments to the dbo ontology.

