

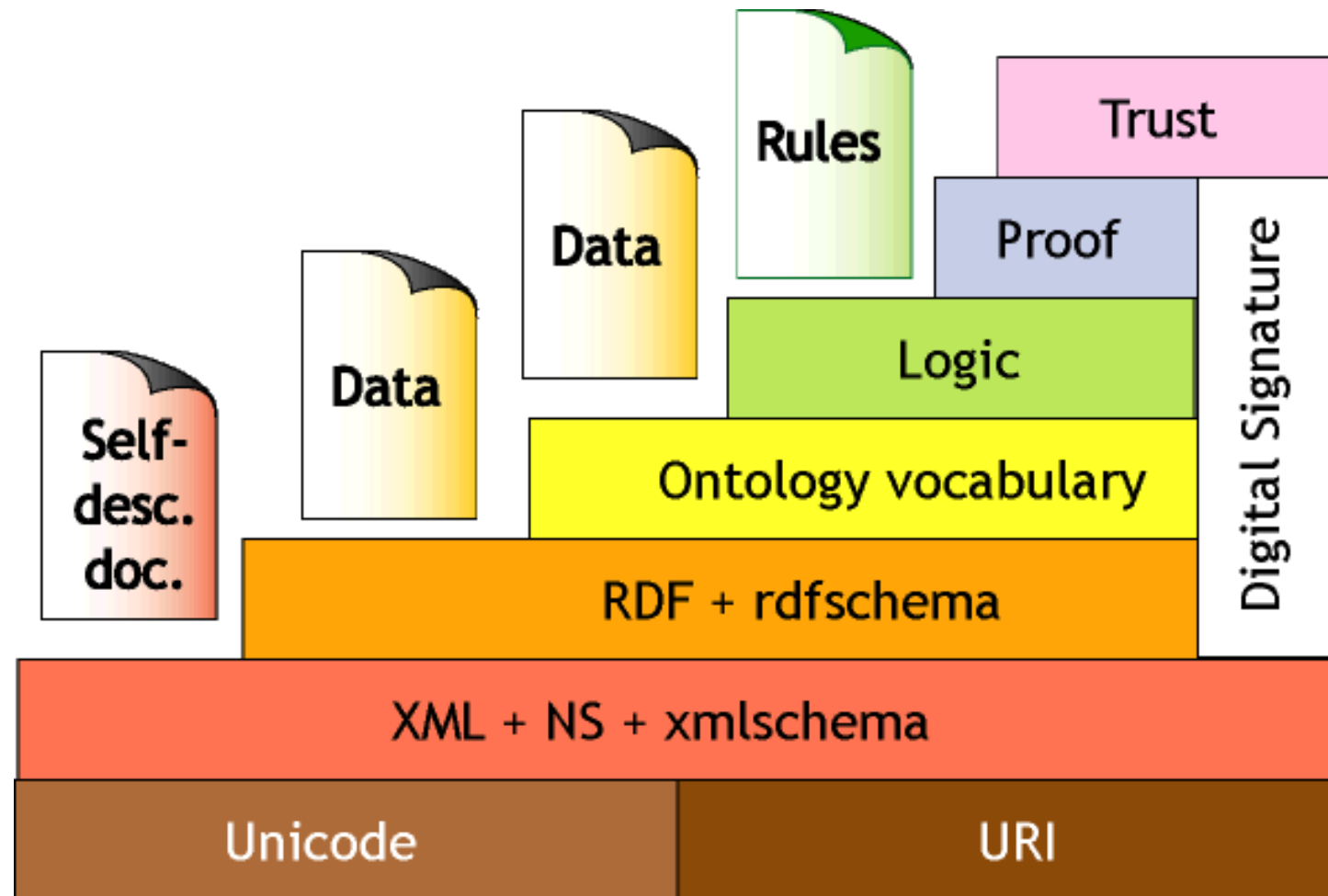


Future Internet

ILARIA TORRE

Ilaria.torre@unige.it

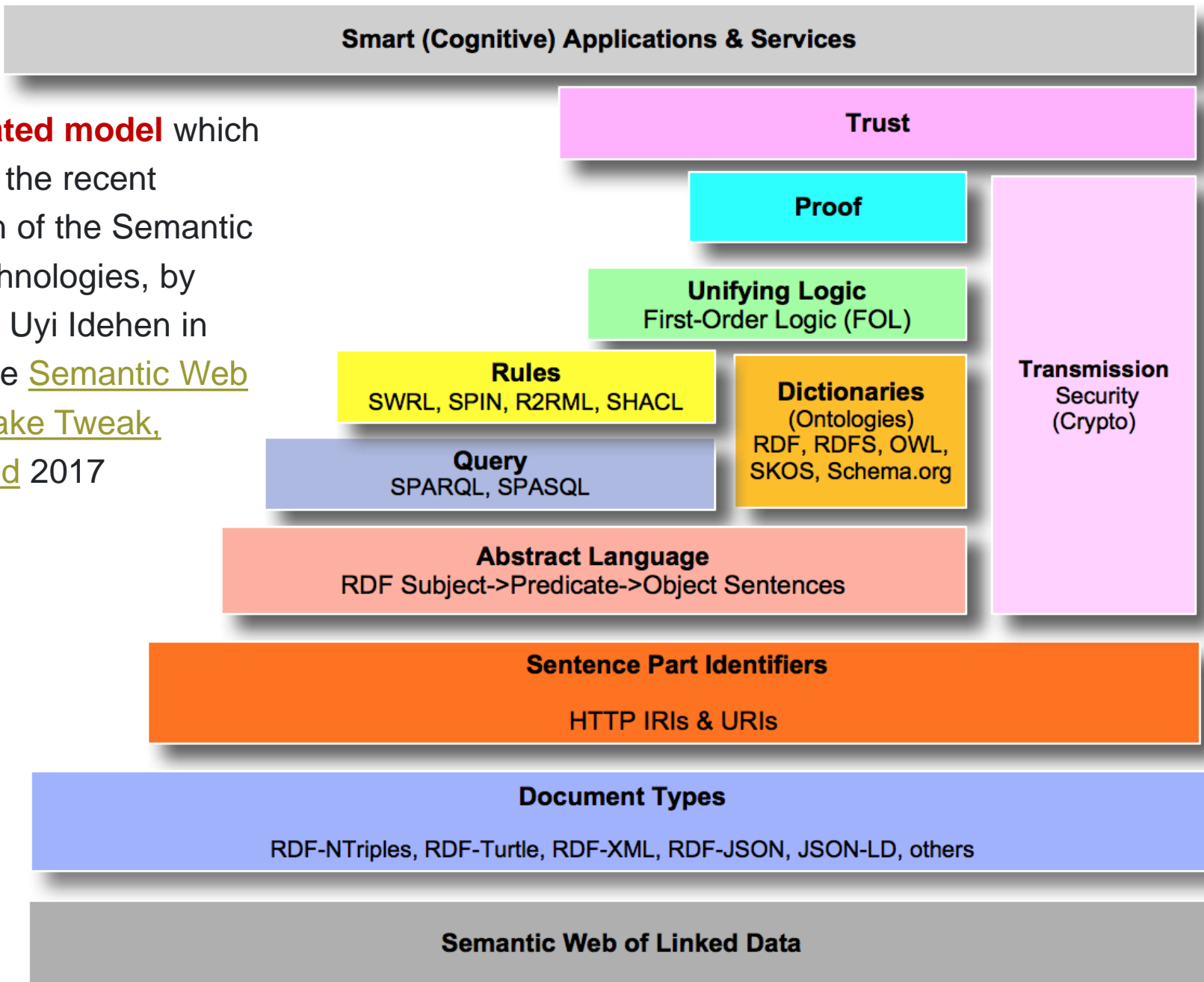
Semantic Web Architecture



Layered Architecture

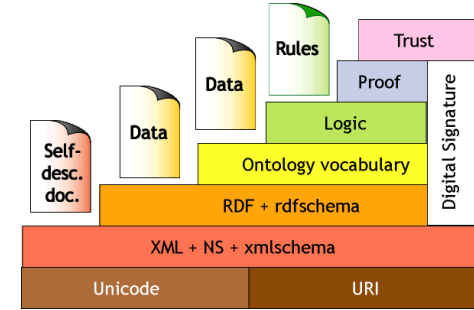
- Layer cake designed by Tim Berners-Lee in 1999
- Each layer uses or extends previous layers
- The initial architecture proposed by Tim Berners-Lee was revised in the following years, but the basic approach remained the same

An **updated model** which includes the recent evolution of the Semantic Web technologies, by Kingsley Uyi Idehen in the article [Semantic Web Layer Cake Tweak, Explained](#) 2017



Architecture

1. Resource level:



UNICODE

- International Standard for Character Encoding
- Universal Character Set (Unicode/ISO 10646)
- It allows you to encode the characters of all languages, mathematical symbols, musical symbols, etc.
- The first 128 characters of UTF-8 (one of UNICODE's encoding schemes) correspond to ASCII encoding

URI

Each resource is identified by **URI**

As Unicode is used → IRI (Internationalized Resource Identifier)

Architettura

2. Mark-up level: XML

XML + NS + xmlschema

XML is an easy way to exchange documents

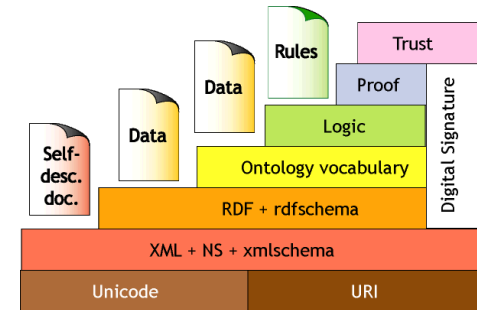
It is machine-readable

It's a meta-language: allows you to

- create the "grammar" and "vocabulary" of a new language (tags and composition rules), specified in a file called **XML Schema Definition**
- associate the XML Schema with a namespace(**NS**) using a URI
- create annotated documents using that language:

XML documents that specify the XML Schema NameSpace, which is the definition of the language used to annotate data.

Note: when the SW architecture was defined by Tim Berners Lee, XML was chosen as the available mark-up language. With the introduction of new mark-up formats, the architecture has accordingly been updated (see previous slide)



Architecture

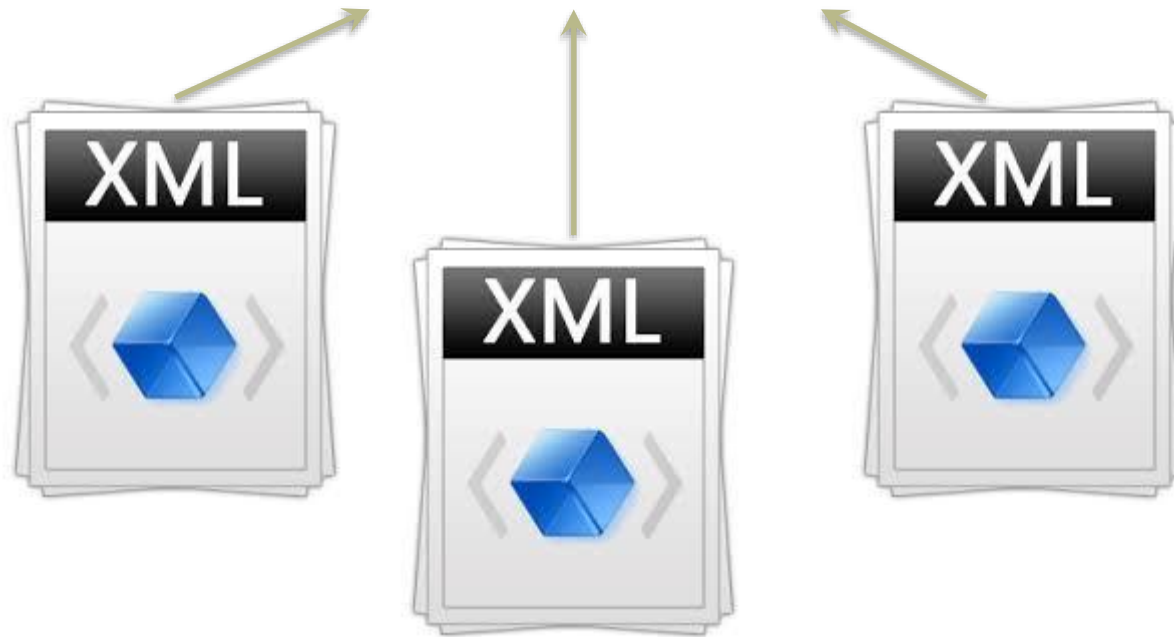
Mark-up level: XML

Example

Dublin Core

XML Schema

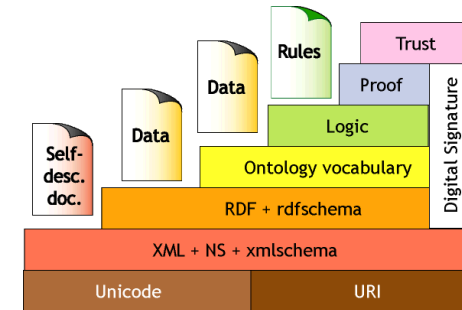
xmlns URI <http://dublincore.org/schemas/xmls/simpledc20021212.xsd>



Architecture

3. Data model layer: RDF

RDF + rdfschema



RDF (Resource Description Framework)

Provides a model and language for defining relationships between resources.

- The **model** defines relationships in terms of "subject-property-object" triples, defining RDF graphs
- The **language** imposes structural constraints in order to provide unambiguous methods for expressing semantic information

RDF/XML has been the first syntax defined by the W3C, to express (i.e. serialize) an RDF graph as an XML document.

RDF Model

- **Resources:** The "things" you are talking about, identified by URIs. (object)
- **Properties:** describe relationships between resources; they are themselves resource types and as such are identified by URIs (attribute: property)
Note: properties can only be binary $p(x,y)$
- **Assertions:** assigning Properties to a Resource. (**value:** the value of a property).

An RDF document is a set of assertions

RDF/XML language: syntax

RDF/XML is sometimes misleadingly called simply RDF as it was historically the first W3C standard RDF serialization format.

- Description of **resources**

<rdf:Description

- **rdf:about**="http://site.com/members" >
- **rdf:ID**="JohnDoe">

- Description of **properties** (relation between resources)

<foaf:knows rdf:resource="#356"

- Specification of the **membership** of a resource in a category (it links a resource to a class).

<rdf:type rdf:resource="person" >

RDFS Language

RDF Schema (RDFS) is a vocabulary for describing properties and resource classes, and provides semantics for constructing a hierarchical structure between them (through the “subClassOf”).

It allows you to express constraints on properties and to validate a document.

RDFS → T-box

RDF → A-box

RDFS/XML Language

namespace for RDFS/XML language

xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

RDF Schema provides a data-modelling vocabulary for RDF data.

*It is a vocabulary to describe **properties** and **classes** of resources and offers semantics to construct a **hierarchical structure** between them (through the property "**subClassOf**").*

*It allows you to express **constraints** on properties and to validate a document.*

RDFS/XML

RDFS allows to:

- define **Classes**
`<rdfs:class rdf:about=professor >`
- define **subclass relationships**
(if A is a subclass of B, then each instance of A is also an instance of B.)
`<rdfs:subClassOf rdf:resource="staff" >`

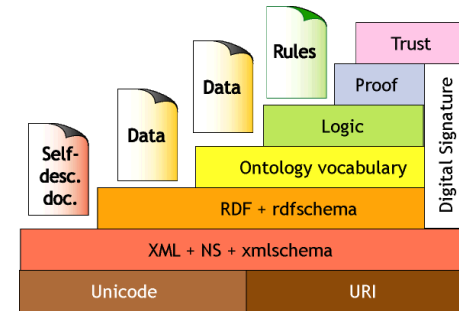
RDFS/XML

- specify **restrictions** on classes in terms of:
 - **range** (class whose instances can appear as values of a triple)
`<rdfs:Property rdf:ID="isTaughtBy" >`
`<rdfs:range rdf:resource="professor" >`
 - **domain** (A class whose instances can be the subject of a triple)
`<rdfs:domain rdf:resource="course">`

Architecture

4. Level of ontologies: OWL

Ontology vocabulary



To formally express ontologies, different languages have been proposed (DAML, OIL, SHOE),

but the current W3C standard is **OWL (Ontology Web Language)**

OWL Language

OWL (Ontology Web Language) allows to

- Specify better **Relationships between Classes**
 - disjunction (males/females)
 - equivalence
 - Boolean combinations
 - complementarity
 - union
 - intersection
- **Restrictions on the properties**
 - about values
 - cardinality

OWL: Syntax

Three versions of OWL have been defined:

- **OWL Full**: Fully RDF compatible; very powerful and expressive; less efficient for reasoning
- **OWL DL** (description logic): Less efficient for reasoning and less compatible with RDF
- **OWL Lite**: restrictions on OWL DL, easy to implement, not very expressive

OWL: Syntax

XML presentation syntax for OWL

- defining a **class**
`<owl:Class rdf:ID="associateProfessor">`
 `<rdfs:subClassOf rdf:resource="staff" >`
`</owl:Class>`
- stating that this class is **disjoint** from another
`<owl:disjointWith rdf:resource="assistantProfessor" >`
- stating that this class is **equivalent** to another
`<owl:equivalentClass rdf:resource="teacher" >`

OWL: Syntax

Properties

- **Object property:** Relates an object to another object ("sub-property")
`<owl:ObjectProperty rdf:ID="isTaughtBy" >`
 `<rdfs:domain rdf:resource="#course">`
 `<rdfs:range rdf:resource="#staff" >`
 `<rdfs:subPropertyOf rdf:resource="involves"`
 `</ owl:ObjectProperty >`
- **Data type property:** Declares a value for a property
`<owl: DataTypeProperty rdf:ID="age" >`
 `<rdfs:range`
 `rdf:resource="www.w3c.org/01/xmlschema#nonNegativeInteger" >`
 `</ owl: DataTypeProperty >`

OWL: Syntax

- **Inverse property**

```
<owl:ObjectProperty rdf:ID="teaches" >  
  <rdfs:domain rdf:resource="staff">  
  <rdfs:range rdf:resource="course" >  
  <owl:inverseOf rdf:resource="isTaughtBy"  
</ owl:ObjectProperty >
```

- **Restrictions on properties**

Some kind of restrictions, including the restriction on possible property values that must come from a specific class

```
<owl:allValueFrom rdf:resource="professor">
```

OWL: Syntax

Boolean operators

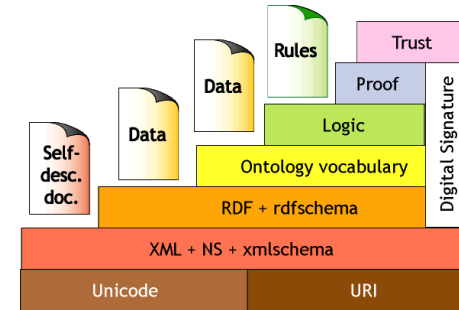
- Complementarity `<owl:ComplementOf >`
- Union between classes `<owl:UnionOf >`
- Intersection `<owl:intersectionOf" >`

Enumeration

```
<owl:OneOf rdf:ID="weekDays" >  
    < owl:thing rdf:about="#Monday">  
    < owl:thing rdf:about="#Tuesday">  
</ owl: OneOf >
```

Architecture

5. Level of reasoning



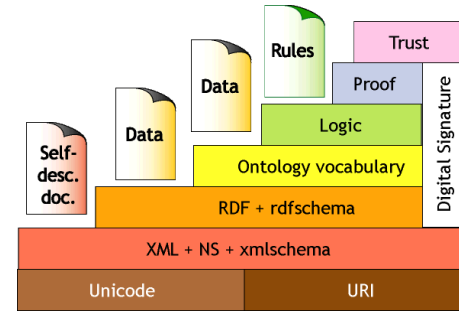
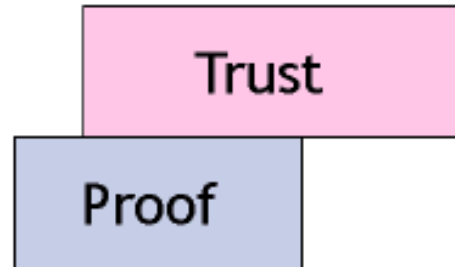
This level aims to enable the writing of rules that use ontology classes and relations to make inferences.

→ Unifying Logic layer

Several rule languages have been defined, but there are currently no W3C recommendations

Architecture

6. Levels of Proof and Trust



The goal of these levels is to define methods for establishing the reliability and veracity of assertions.

Currently, however, no standards have yet been defined

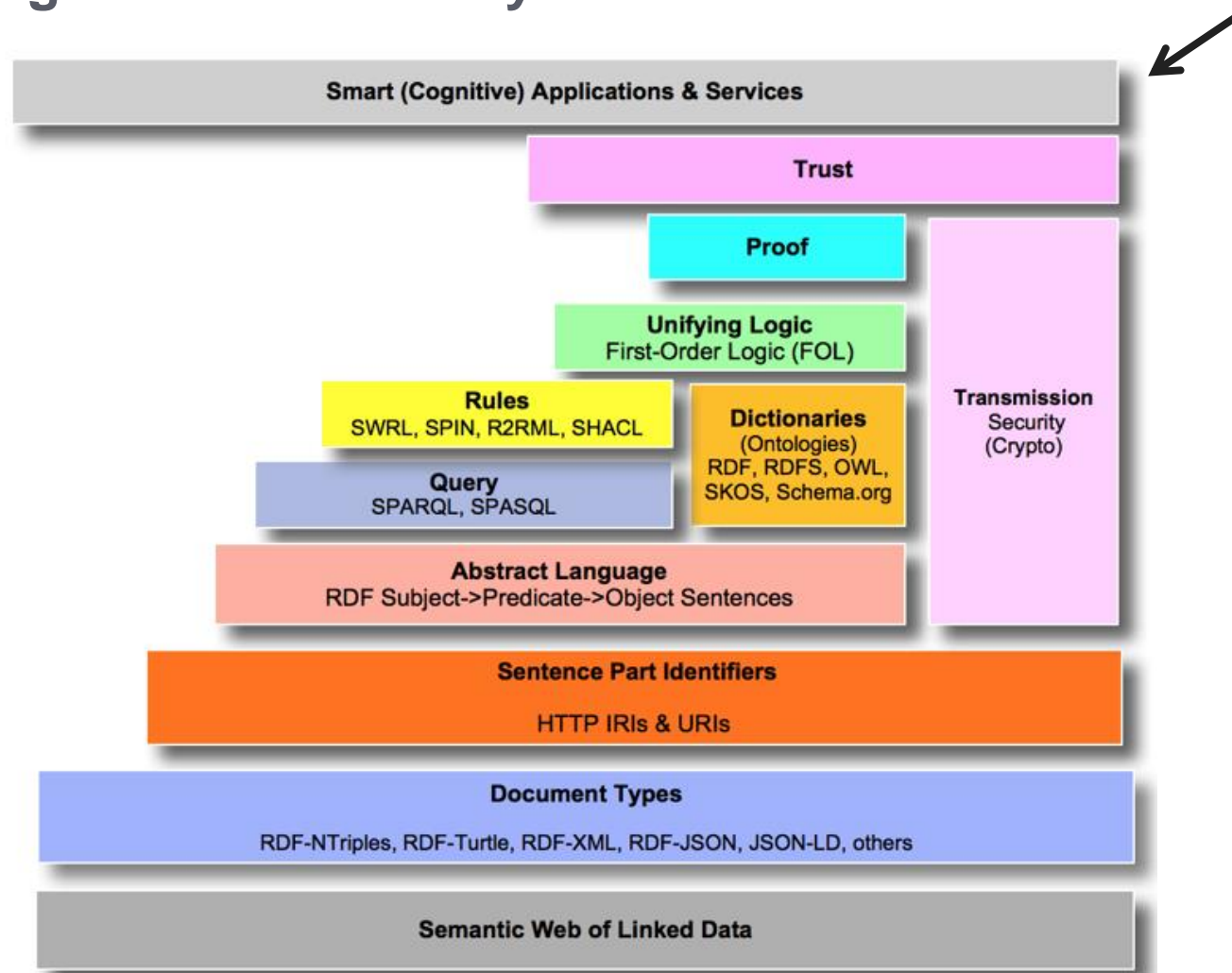
Digital Signature and **Encryption** prove the actual authorship of an assertion; It will be up to the user to decide which digital signatures to trust.

→ objective “**Web of Trust**”: If trust = transitive property, users can be tied into a web of trust

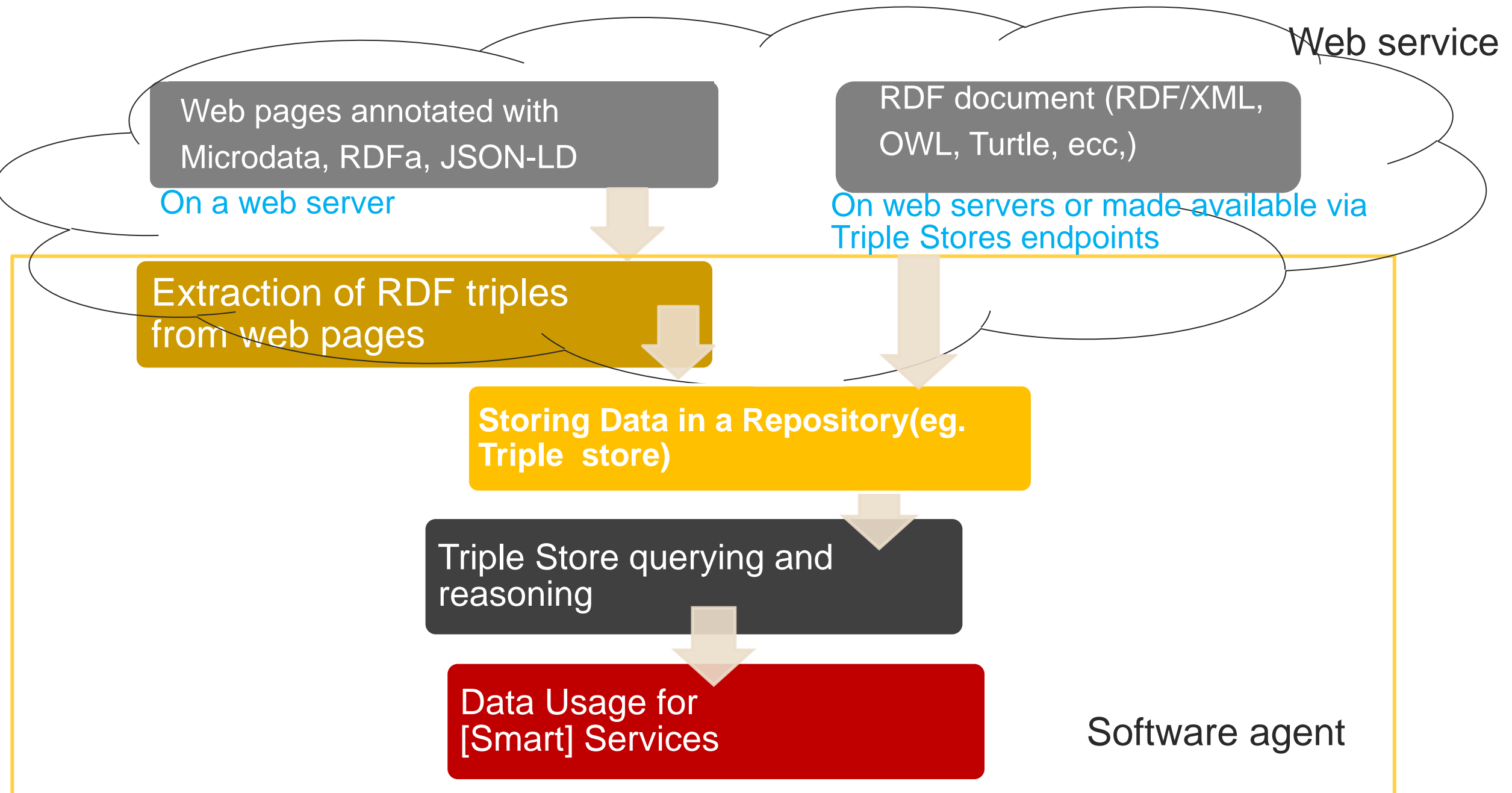
Architecture

Smart (Cognitive) Applications and Services

Working with semantically annotated data



RDF TRIPLES extraction and use



Use of semantically annotated data

- ❑ Top of figure: Data semantically annotated and made available on the Web
 - Web pages annotated using JSON-LD, RDFa, and Microdata
 - RDF documents annotated using specific formats for the Semantic Web (RDF/xml, Turtle, ecc.)

- ❑ Bottom of figure: Software agents access annotated data for use in other services.

Software agents can search for data based on its content, compare data, integrate data, reason on data to infer new data, make diagnoses, predictive analysis, etc.

Use of semantically annotated data

The acquired data can be saved by the sw agent in repositories.

Data storage can be done directly in the file system, or in **TRIPLE stores**, also called RDF databases (an example is Virtuoso).

Data can also be transformed and stored in DB.

Finally, they are used to provide other services that, typically, require the integration of data extracted from different sources and may also employ reasoning mechanisms on the semantics of the data itself, for example to make inferences.

NB: TRIPLE stores can be queried in a similar way to how a database is queried. The standard language for querying RDF is **SPARQL**

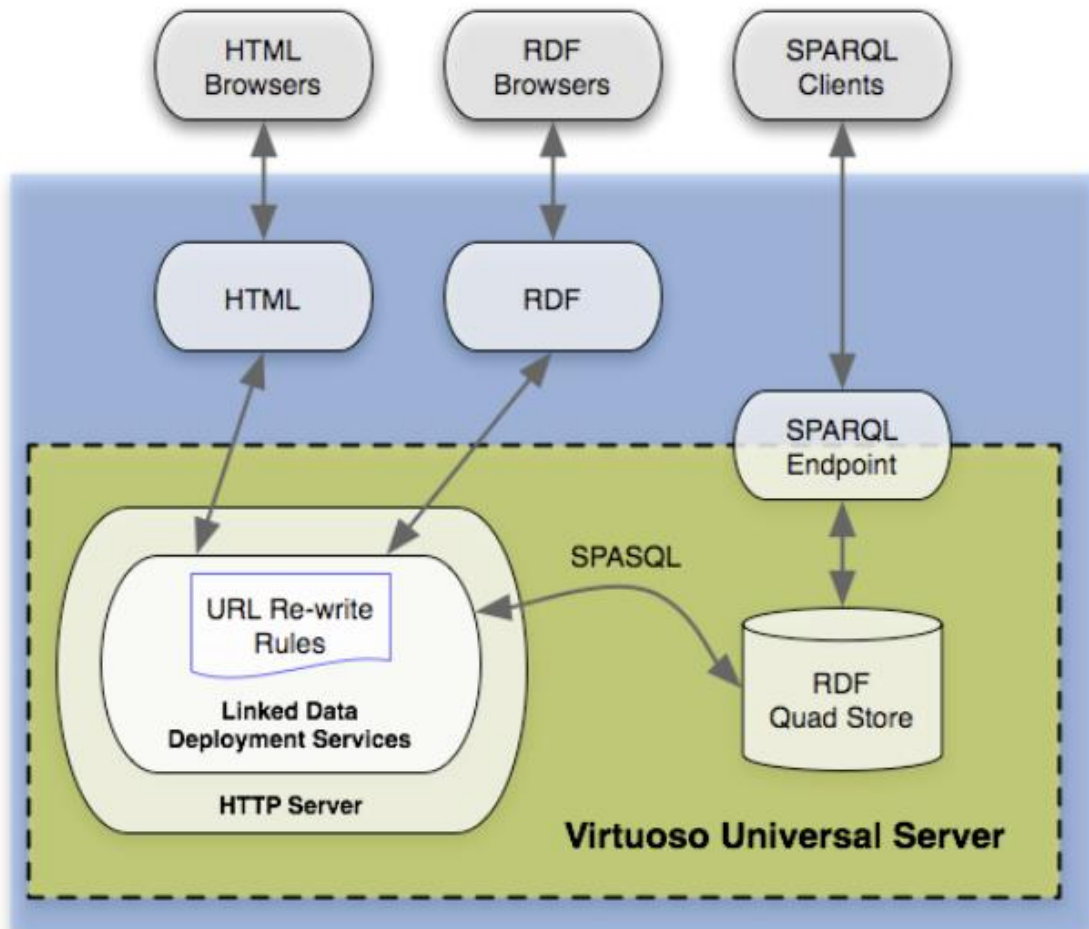
Triple extraction from Web pages

For web pages annotated with JSON-LD, RDFa, and Microdata, an intermediate step of triple extraction is required for the software agent to use the data(see previous figure) in order to obtain clean **RDF TRIPLES** (=without the HTML layout and presentation elements)

Virtuoso

One of the most used triple stores is Virtuoso.

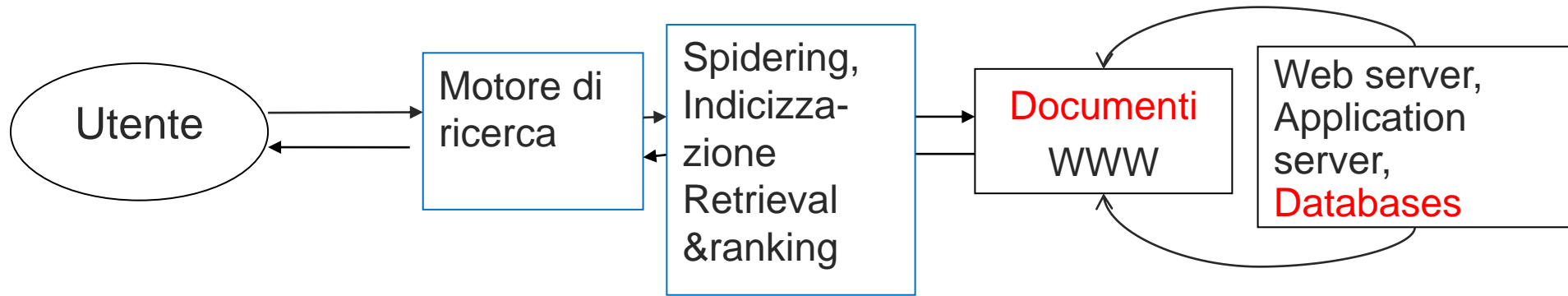
Virtuoso is also used by **DBpedia**, the RDF version of Wikipedia



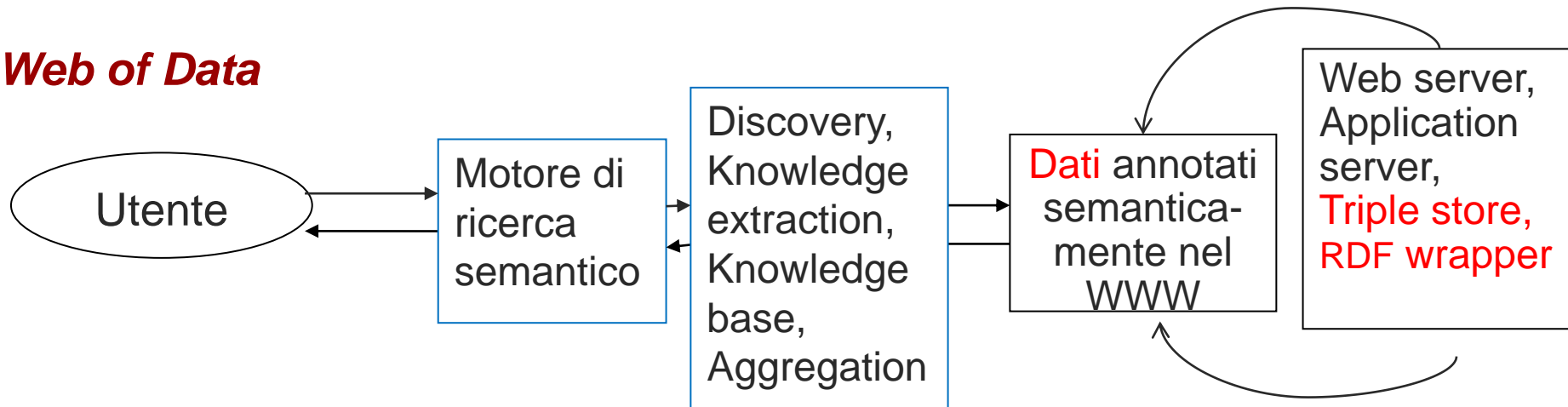
Applications of the Semantic Web

An example

Web



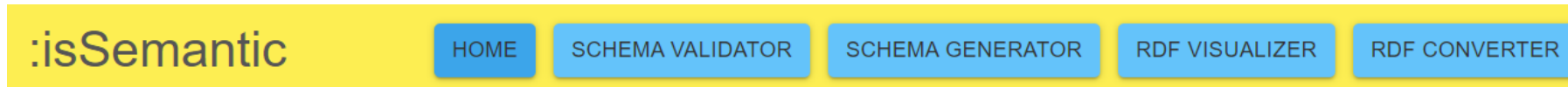
Web of Data



Extracting, Converting, and Displaying RDF Triples

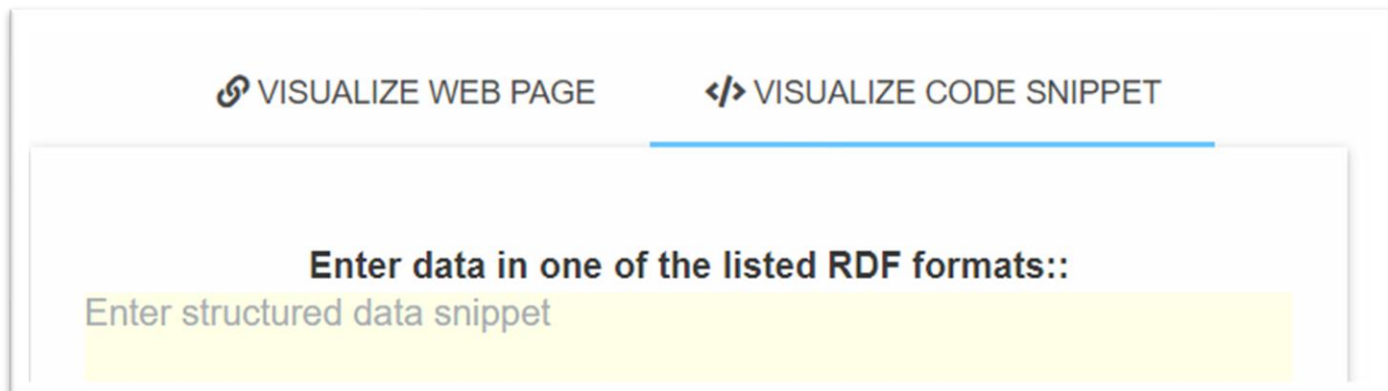
There are services to extract, convert, display RDF graphs in different serialization languages.

For example, **isSemantic** <https://issemantic.net/>



isSemantic.net is a structured data testing tool that allows you to validate schema markup, rich snippets, Open Graph and Twitter Cards tags, visualize, generate and convert schema markup and structured data.


Note. Beware of advertisements: you don't need to sign up, just enter the code to be converted/displayed in the space provided

The image shows a screenshot of the 'VISUALIZE CODE SNIPPET' section of the isSemantic tool. At the top, there are two tabs: 'VISUALIZE WEB PAGE' and 'VISUALIZE CODE SNIPPET', with the latter being selected. Below the tabs, there is a text input area with the prompt 'Enter data in one of the listed RDF formats::' and a yellow highlighted box containing the text 'Enter structured data snippet'.

Microdata to RDF Distiller

Example: Automatic extraction of RDF triples from HTML files annotated with microdata

```
<section itemscope itemtype="http://schema.org/Person">
  Hello, my name is
  <span itemprop="name">John Doe</span>,
  I am a <span itemprop="jobTitle">graduate research
  assistant</span>
  at the <span itemprop="affiliation">University of
  Dreams</span>.
  My friends call me <span
  itemprop="additionalName">Johnny</span>.
  You can visit my homepage at
  <a href=http://www.JohnnyD.com
  itemprop="url">www.JohnnyD.com</a>.
  <div itemprop="address" itemscope
    itemtype="http://schema.org/PostalAddress">
    I live at <span itemprop="streetAddress">1234
    Peach Drive</span>,
    <span itemprop="addressLocality">Warner
    Robins</span>,
    <span itemprop="addressRegion">Georgia</span>.
  </div>
</section>
```




```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:schema="http://schema.org/"
>
  <Person>
    <url rdf:resource="http://www.JohnnyD.com"/>
    <affiliation>University of Dreams</affiliation>
    <jobTitle>graduate research assistant</jobTitle>
    <additionalName>Johnny</additionalName>
    <name>John Doe</name>
    <address>
      <PostalAddress>
        <streetAddress>1234 Peach Drive</streetAddress>
        <addressRegion>Georgia</addressRegion>
        <addressLocality>Warner Robins</addressLocality>
      </PostalAddress>
    </address>
  </Person>
</rdf:RDF>
```


RDFa 1.1 Distiller and Parser

Example: Automatic extraction of RDF triples from HTML files annotated with RDFa

```
<section vocab="http://schema.org/" typeof="Person">
  Hello, my name is <span property="name">John
  Doe</span>,
  I am a <span property="jobTitle">graduate research
  assistant</span> at the <span
  property="affiliation">University of Dreams</span>.
  My friends call me <span
  property="additionalName">Johnny</span>.
  You can visit my homepage at
  <a property="url"
  href="http://www.JohnnyD.com">www.JohnnyD.com</a>
  <div property="address">I live at
    <span typeof="PostalAddress">
      <span property="streetAddress">1234 Peach
      Drive</span>,
      <span property="addressLocality">Warner
      Robins</span>,
      <span property="addressRegion">Georgia</span>.
    </span>
  </div>
</section>
```



```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:schema="http://schema.org/"
>
  <Person>
    <url rdf:resource="http://www.JohnnyD.com"/>
    <affiliation>University of Dreams</affiliation>
    <jobTitle>graduate research assistant</jobTitle>
    <additionalName>Johnny</additionalName>
    <name>John Doe</name>
    <address>
      <PostalAddress>
        <streetAddress>1234 Peach Drive</streetAddress>
        <addressRegion>Georgia</addressRegion>
        <addressLocality>Warner Robins</addressLocality>
      </PostalAddress>
    </address>
  </Person>
</rdf:RDF>
```