Self-explaining APIs

A machine-readable, semantic approach to schema design

EuroPython
Dublin, 14th July 2022

Roberto Polli
API EXPERT

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

EuroPython
2022 DUBLIN & REMOTE

Agenda

A semantic approach to schema design improves API interoperability and integration

- **Understandable API messages**
- **Controlled Vocabularies**
- **Contract-first schema design**
- **Central catalog for semantic asset**

Audience: API designers and developers familiar with OpenAPI, JSON Schema, XMLSchema and YAML

Caveat: Not a rigorous / theoretical definition of semantic web concepts

Simplify the creation of API mash-up

Uniform the meaning and syntax of APIs produced by thousands of providers

Semantics, why should I care?

Semantics is the study of meaning, and ensures that **messages** are understood. Messages includes HTTP exchanges and data content.

Two ambiguous API messages

➔ `{ name: "FABIANO Romildo" }`

Is that a given name, a full name or a birth name? Which is the given name?

➔
```
{ name: "FABIANO Romildo"
  income: 4_000_000     }
```

Which is the currency?

We can't mash-up APIs with different formats and meanings!

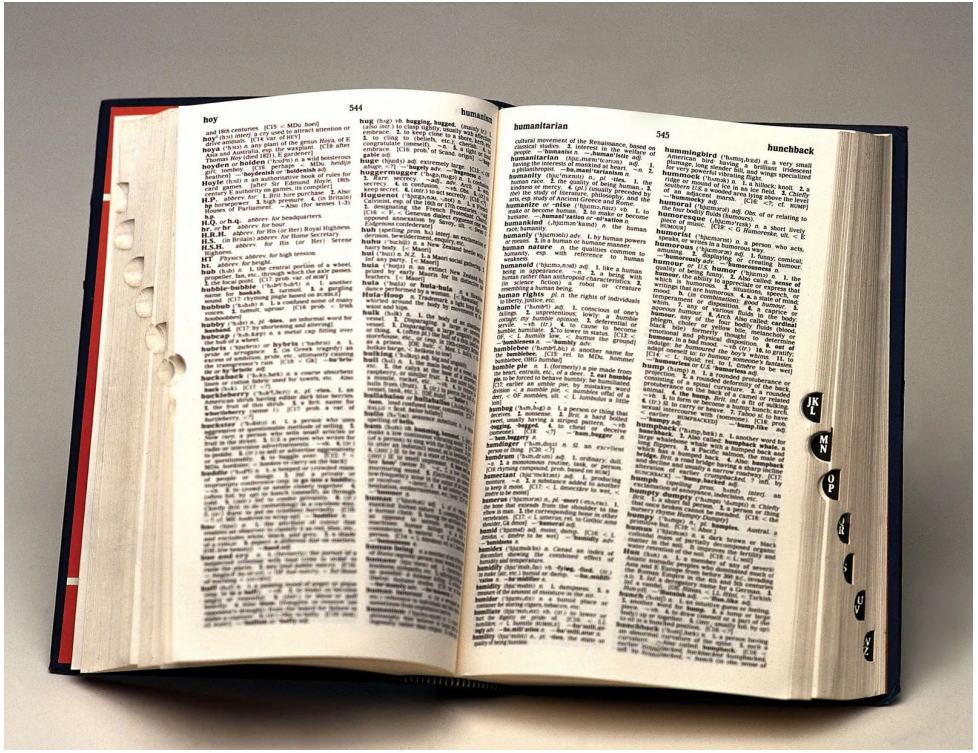
Vocabularies to the rescue

Controlled Vocabularies use URIs for disambiguating terms and provide semantics

Every **term** is identified by an absolute URI. The prefix identifies the vocabulary name

https://dbpedia.org/data/ **Dog** 'The dog is a four leg animal'

vocabulary name (prefix) term definition (rdfs:comment)



Vocabularies to the rescue

All kinds of controlled vocabularies are formally described by formats like JSON-LD or Turtle. Those formats are equivalent.

Vocabularies:

- define concepts and relationships in a specific domain (e.g. healthcare, finance, personal information, ...)
- validated by a designated authority
- formally described using **`text/turtle`** or its JSON counterpart: **`application/ld+json`** (JSON Linked Data W3C specification)

Some vocabulary types:

- **ontologies**: complex and formal collections of terms
- **code lists**: simple lists of terms
- **taxonomies**: hierarchical structure

Discover the European Union controlled vocabularies!

Vocabularies in practice

Four vocabulary terms described in Turtle format

Declare the URI namespaces of the vocabularies we are using

Define terms using **one or more sentences**

A sentence is formed by **a triple ending with a dot**

subject predicate object .

Spaces are not relevant :)

```
# Reuse the rdf-schema terms.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

# Declare the namespace for the Italian Core Person Vocabulary.
@prefix IT: <https://w3id.org/italia/onto/CPV/> .

# Define the meaning of three terms.
IT:Person            rdfs:comment   """A natural person
                                      (e.g. not a company)."""
.
IT:givenName         rdfs:comment """The given name of a person.
                                     E.g. 'Anna'..."""
.
IT:RegisteredFamily  rdfs:comment """[..] a group of people tied together
                                     [..according to ..] (Italian Law DPR
                                     233/1989 art.4)"""
.
# Two sentences on the same subject, which is a relation between persons
IT:isChildOf         rdfs:comment """The child-parent relation."""
.
IT:isChildOf         rdfs:domain   IT:Person
.
```

Vocabularies in practice

Use rdflib to process Turtle files, and convert them in other formats.

Turtle files are loaded into a graph.

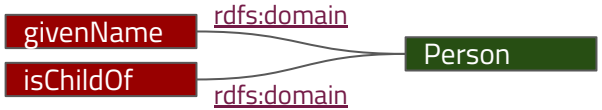
A graph can be serialized in other formats, including JSON-LD and XML.

JSON LD represents subjects as objects and predicates as keys.

```
from rdflib import Graph    # pip install rdflib

# Add the vocabularies to the namespaces
context = { "IT": "https://w3id.org/italia/onto/CPV/",
            "rdfs": "http://www.w3.org/2000/01/rdf-schema#"}

# Parse the text in the previous slide in a Graph
vocab = Graph()
vocab.parse(data=turtle_text, format="text/turtle")

# Serialize to JSON-LD
json_text = vocab.serialize(format="application/ld+json", context=context)

{ "@context":  { "IT": "https://w3id.org/italia/onto/CPV/", ..}
  "@graph": [  {  "@id": "IT:isChildOf",
                  "rdfs:comment": "The child-parent relation",
                  "rdfs:domain":     {"@id": "IT:Person }
                }, { ... other terms, ...}


vocab_json = json.loads(json_text)
```

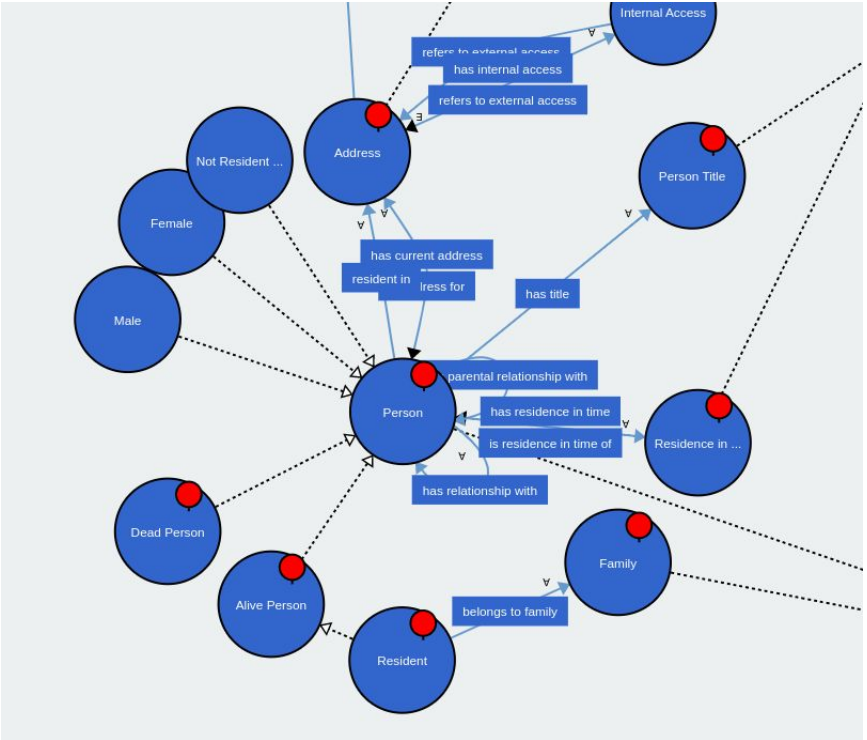
Vocabularies in practice

A graph representation of a vocabulary: subjects and objects are connected by predicates.



Predicates and objects can be subject of other sentences, thus creating "Knowledge Graphs".

The figure shows a simplified graph of the entities described in the Italian vocabulary for person



Example: the EU vocabulary for countries

Based on SKOS and Dublin Core standard vocabularies used for defining vocabularies.

Syntax support internationalization using language tags.

Sentences with the same subject / predicate can be shortened using ";" and ","

Can correlate terms, like older and newer countries.

Vocabularies improve data quality.

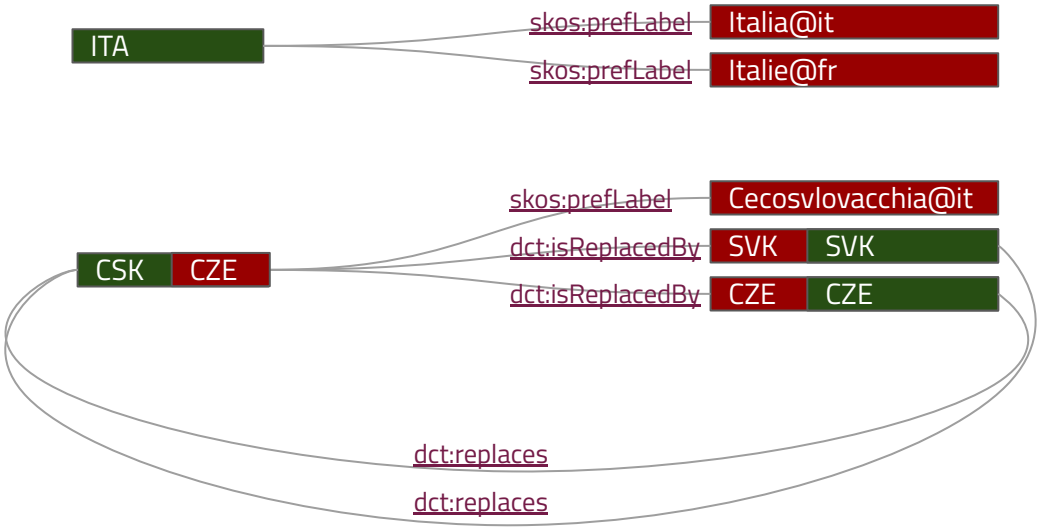
```
@prefix country: <https://publications.europa.eu/resource/authority/country/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dct: <http://purl.org/dc/terms/> .

country:ITA  dc:identifier ITA  .
country:ITA  skos:prefLabel "Italie"@fr .
country:ITA  skos:prefLabel "Italia"@it .

# A shorter syntax for the above 3 sentences
country:FRA dc:identifier       FRA;
            skos:prefLabel    "France"@fr, "Francia"@it ; ...
.

# Correlating terms directly (dct:replaces)  and inversely (dct:isReplacedBy)
country:CZE dct:replaces        country:CSK ; ..
.
country:SVK dct:replaces        country:CSK ; ..
.
country:CSK dct:isReplacedBy country:CZE ;
            dct:isReplacedBy country:SVK ; ..
.
```

The EU vocabulary for countries

This vocabulary is more than a codelist: a country can be a subject in a sentence, and an object in another.

In the image, this is rendered using both red and green.

Predicates can be both objects and subjects too!



Vocabularies are stored in graph databases

- Stored in knowledge graph databases (e.g. Virtuoso, Amazon Neptune, ...)
- Queried using the SparQL protocol (e.g. on https://data.europa.eu/sparql)

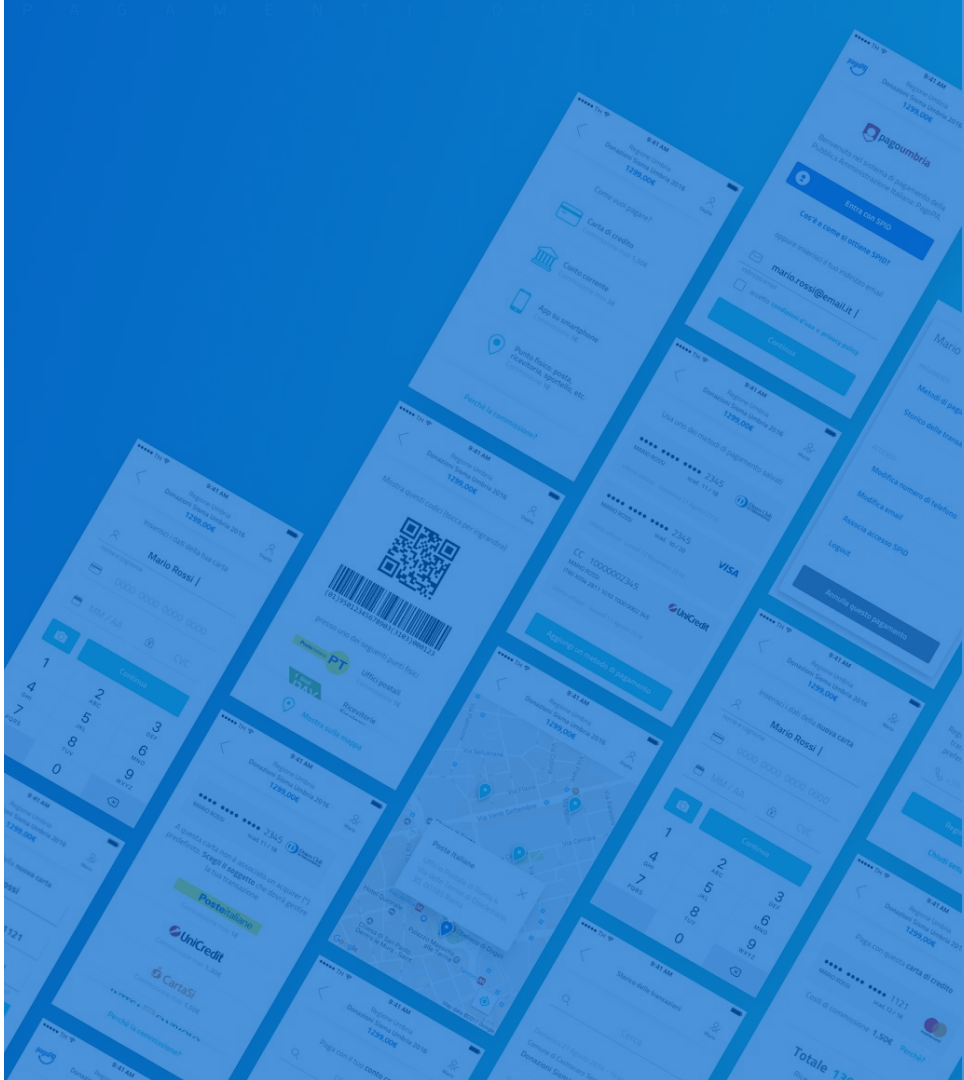
```
PREFIX euvoc: <http://publications.europa.eu/resource/authority/>
...
SELECT ?URI, ?concept, ?identifier
WHERE {
  ?URI       skos:inScheme  euvoc:country ;
             skos:prefLabel ?concept      ;
             dc:identifier  ?identifier
  FILTER(lang(?concept) = 'en')
}
```

URI	concept	identifier
https://publications.europa.eu/resource/authority/country/ITA	Italy	ITA
https://publications.europa.eu/resource/authority/country/FRA	France	FRA

try me via bit.ly

Using vocabularies

Vocabularies contain a lot of information. There are tools and specifications to present views of those resources in different formats.



Describing data with vocabularies

An example of data described using vocabularies.

Identified by an email URI

Defined by 5 sentences

Predicates are imported from well defined terms from the @prefix'ed vocabularies.

Applications can use this information to automate interoperability checks and other logics.

```
@prefix country: <https://publications.europa.eu/resource/authority/country/> .
@prefix dc:      <http://purl.org/dc/elements/1.1/> .
@prefix IT: <https://w3id.org/italia/onto/CPV/> .

<mailto:robipolli@gmail.com>                # A subject identified by an email.
  a                    IT:Person ;          # it is a Person
  IT:givenName    "Roberto" ;
  IT:familyName   "Polli" ;
  IT:hasBirthPlace country:ITA;             # birthplace uses the EU vocabulary
  dc:identifier        "robipolli@gmail.com"; ...
.
```

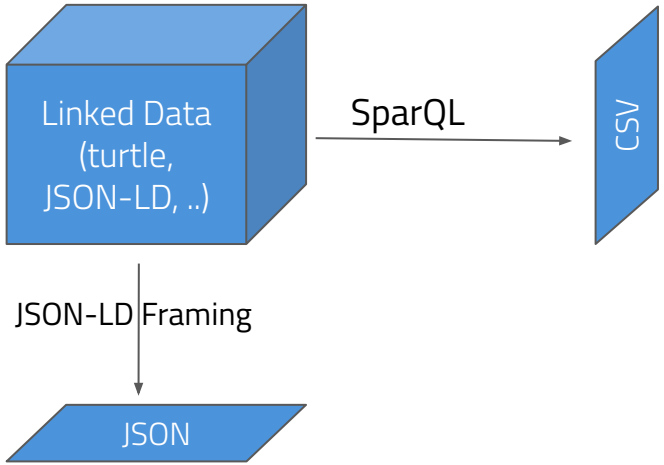
Benefits of Linked Data

Data described in Turtle or JSON-LD can be projected to simpler representations, like CSV or JSON using standard specifications

JSON-LD Framing is a W3C specification that allow to project Linked Data to JSON with lower dimensions specifying an object called @context that can be used to reverse the transformation.

SparQL queries are flexible, but hardly reversible.

CSV-W (aka rdf-tabular) is a specification that allows to annotate CSV files with semantic information



JSON-LD Framing

Match and transform JSON-LD subjects to create a JSON object with the same structure as the **frame**

A subject is selected using the values in the frame.

@context provides information to map predicates and objects to JSON keywords and values.

@context allows to provide semantic information to interpret the resulting JSON.

```
from rdflib import Graph    # pip install rdflib pyld
from pyld import jsonld

countries = Graph()
countries.parse("https://bit.ly/39ZiZZy")
countries_tree = json.loads(countries.serialize(format="application/ld+json"))

# Define a projection
countries_json = jsonld.frame(countries_tree, frame={
  "@explicit": True,                # Select only mentioned keywords
  "@type":  ["skos:Concept"],  # Selects only subjects of type skos:Concept
  "country_code": {},
  "version_info": {},
  "label_en": {},
  "@context": {
      "dc": "http://purl.org/dc/elements/1.1/",
      "skos": "http://www.w3.org/2004/02/skos/core#",
      "owl": "http://www.w3.org/2002/07/owl#",
      "url": "@id",
      "country_code": "dc:identifier",
      "version_info":   "owl:versionInfo",
      "label_en":         {"@id": "skos:prefLabel", "@language": "en"},
  }
})
```

JSON-LD Framing example

```
@prefix country: <https://publications.europa.eu/resource/authority/country/> .
@prefix lang: <https://publications.europa.eu/resource/authority/language/> .
...
@prefix ev: <http://publications.europa.eu/ontology/euvoc#>

country:IRL
 dc:identifier "IRL" ;
 rdf:type skos:Concept  ;

 dct:language [ lang:GLE, lang:ENG ] ;
 ... many other predicates ... ;

 skos:prefLabel "Irlanda"@it,
                "Ireland"@en,
                "Irlande"@fr ;

 owl:versionInfo "20220316-0" ;

 ev:euroCurrencyAdoptionDate "1999-01-01"^^xs:date ;
.
...
```

```
# Frame
@explicit: True
@type: [ skos:Concept ]
country_code: {}
version_info: {}
label_en: {}
# Context
@context:
   url: @id
   label_it:
      @language: it
      @id: skos:prefLabel

   country_code: dc:identifier
   version_info: owl:versionInfo
```

```
{ "@graph": [{
    # url is populated with the resource URI
    "url": "https://...europa.eu/../country/IRL",

    # Only subjects of type skos:Concept
    # will be processed
    "@type": "skos:Concept",
    "country_code": "IRL",

    # label_it is populated with only the Italian
    #   localization of the skos:prefLabel
    #  predicate's  object
    "label_it": "Irlanda",

    # version_info is populated with the object
    #   of the owl:versionInfo   predicate
    "version_info": "20220316-0"
   }, {.. other entries .. }
],
"@context": { ...  same as the frame's @context ... }
 }
```

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE

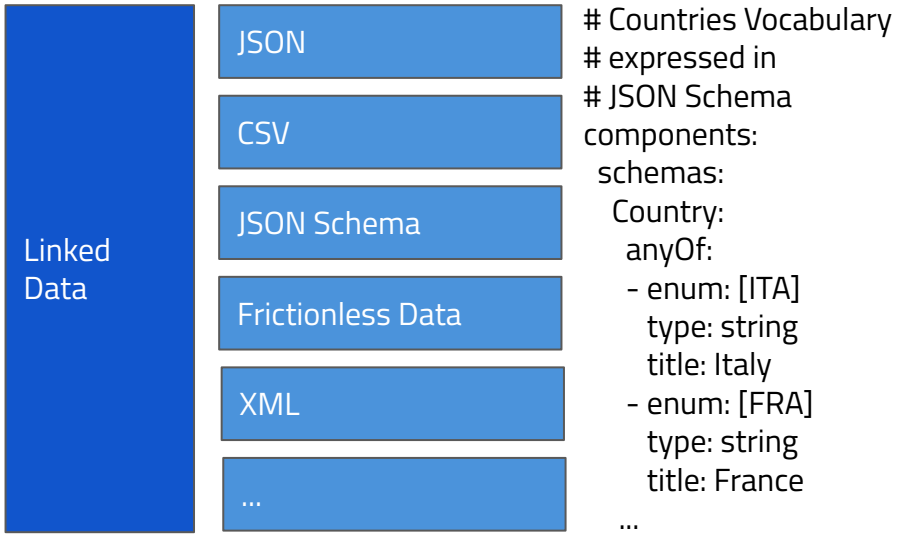
Making data accessible

Linked Data platforms can publish data in different formats to make them usable directly in APIs or in online fillable forms.

Publishing vocabularies in various formats ease their usage: developers don't even need to know that they are using semantic assets!

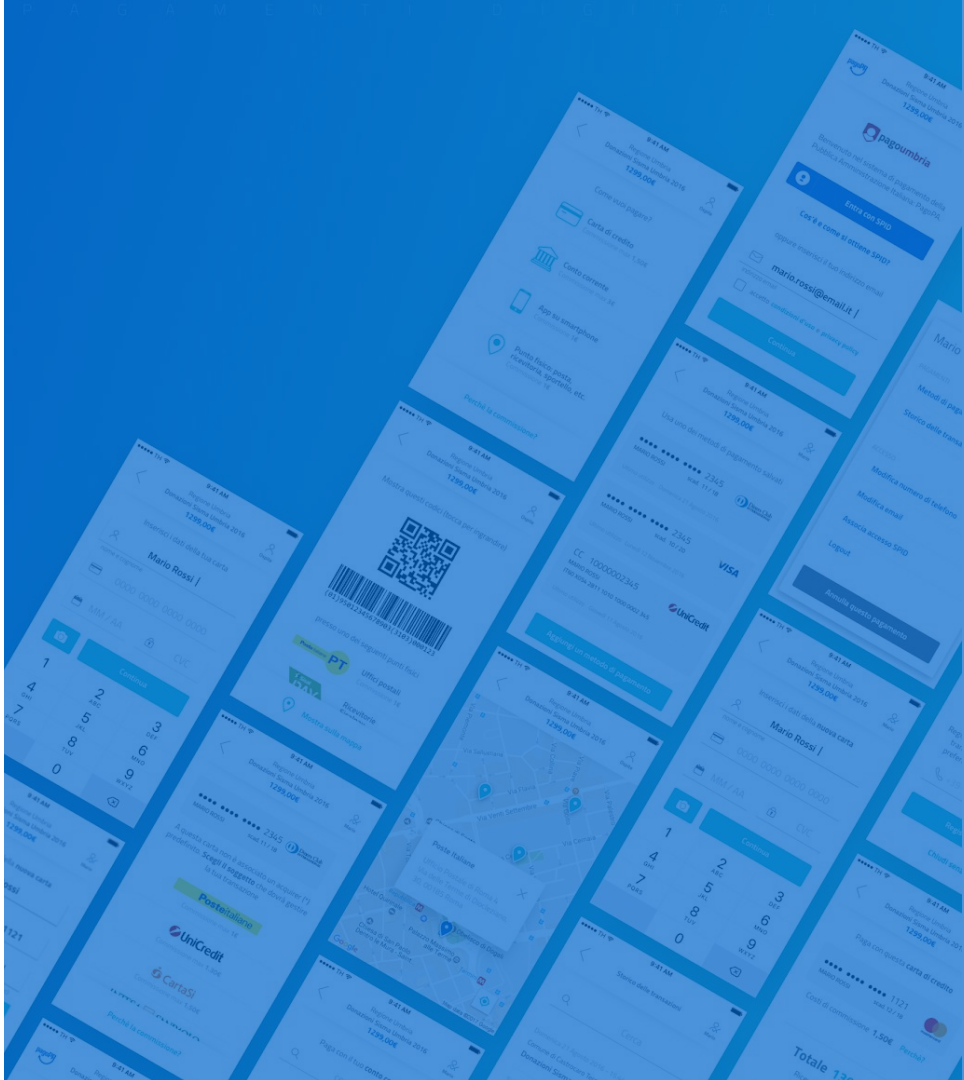
For example, you can pack the data as a JSON Schema file to be included in an API, or in a Frictionless Data Package.

The important part is to always be able to map back the data to the original linked data information.



Semantic APIs

APIs can reference concepts and vocabularies to provide a complete and machine readable description of the exchanged contents



Semantic APIs use the same Vocabularies

When **different APIs use the same vocabularies**, **JSON-LD @context** allows to map JSON properties to vocabulary terms

 # This is an API payload.
{ nome_proprio: Mario, cittadinanza: ITA }

```
# Map object properties to vocabulary terms.
@context:
  nome_proprio: 'https://w3id.org/italia/onto/CPV/givenName'
  cittadinanza:
    @id: 'https://w3id.org/italia/onto/CPV/hasCitizenship'
    @context:
      @base: 'http://...europa.eu/../country/'
    @type: @id
```

 # This is another API payload.
{ given_name: Mario, citizenship: ITA }

```
# Map object properties to vocabulary terms.
@context:
  given_name: 'https://w3id.org/italia/onto/CPV/givenName'
  citizenship:
    @id: 'https://w3id.org/italia/onto/CPV/hasCitizenship'
    @context:
      @base: 'http://....europa.eu/../country/'
    @type: @id
```

 # Different JSON objects map to the same information
 # Country references a codelist

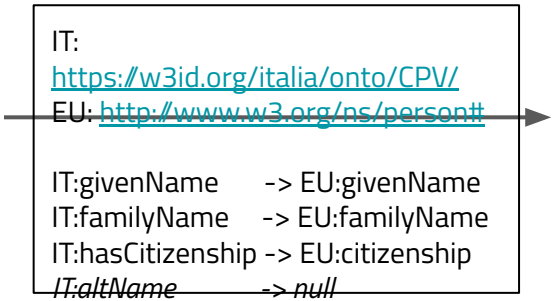
```
@prefix IT: <https://w3id.org/italia/onto/CPV/> .
_:user IT:givenName "Mario"
_:user IT:hasCitizenship <http://...europa.eu/../country/ITA>
```

Semantic APIs use the same Vocabularies

Vocabulary mappings allow to translate objects between different vocabularies.
EU countries have vocabularies e.g. 🇮🇹 → 🇪🇺 → 🇫🇮 .
Interoperability is limited to the subset of compatible fields.

{ nome_proprio: Valentina,
secondo_nome: Maria Elisa,
cognome: Rossi,
paese: ITA }

@context:
IT: https://w3id.org/italia/onto/CPV/
nome_proprio: IT:givenName
cognome: IT:familyName
paese: IT:hasCitizenship
secondo_nome: IT:altName
...



{ givenName: Valentina,
familyName: Rossi,
citizenship: ITA }

@context:
EU: https://w3.org/ns/person#
givenName: EU:givenName
familyName: EU:familyName
citizenship: EU:citizenship
...

JSON LD and OpenAPI: a proposal

Design your schema accurately using a contract-first approach. ①
Associate properties with vocabularies as much as possible: this does not require using identical field names.

Include examples to test your design. ②

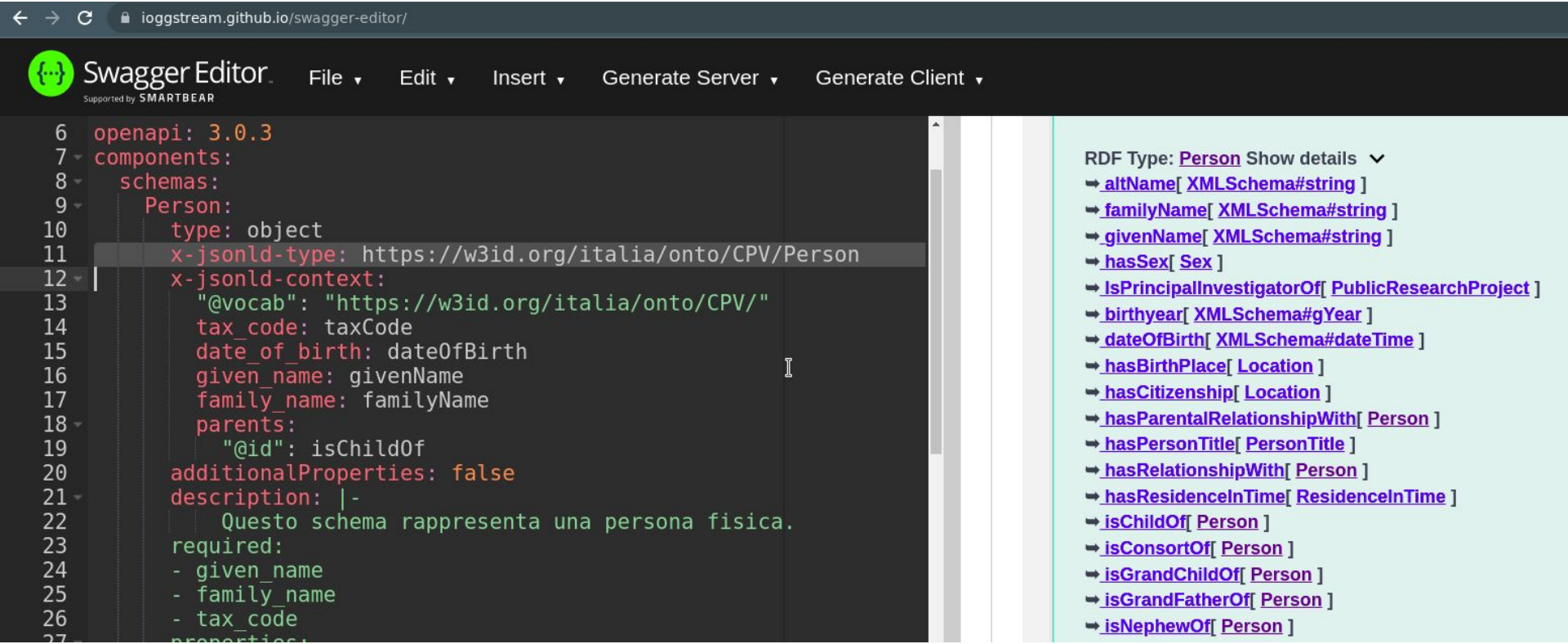
Embed semantic mapping information in OpenAPI / JSON Schema. ③

There's no standard to embed these information in OAS3.0 yet, so we introduce extension keywords like **x-jsonld-context** in this Draft RFC.

Person: # Open in editor

```
# ① A standard OAS3.0 schema (json-schema draft-4)
 type: object
 required: [givenName, familyName, country]
 properties:
   familyName: { maxLength: 255, type: string }
   givenName:  { maxLength: 255, type: string }
   country:    { maxlength: 3, minLength: 3,  type: string }
# ② Example data
example: { familyName: Rossi, givenName: Mario, country: ITA }

# ③ A JSON-LD @context for semantic definitions
x-jsonld-context:
  "@vocab": " https://w3.org/ns/person#"
  country:
    "@id": citizenship
    "@type": "@id"
    "@context":
      "@base": "http://...europa.eu/../country/"
```

Semantic API editor Proof of Concept



National Data Catalog for Semantic interoperability

Italy is working on a National Data Catalog for Semantic interoperability

Italy already provides a set of Controlled Vocabularies at https://w3id.org/italia.

Sources are on github.

Vocabularies are aligned with European Authority Tables.

We are working on a new National Data Catalog for Semantic Interoperability to improve current assets and ease their use:

- find reusable vocabularies and ontologies
- share semantically interoperable schemas for public services
- ensure that APIs have a coherent meaning
- publish relevant codelists (e.g. provinces, cities, document types, ..) via API in easy-to-use formats like OpenAPI or JSON Schema

Semantic interoperability specs

Interoperability requires specifications, so we are engaging various communities to achieve these goals!

Media Type registrations

- YAML (IETF) media type has not been formalized. This specification registers the application/yaml media type and provides security and interoperability considerations.
- YAML-LD (W3C) data format allows to serialize Linked Data in YAML, thus extending JSON-LD. It aims at registering the application/ld+yaml too.
- OpenAPI & JSON Schema (IETF) wants to register openapi+json and schema+json media type providing interoperability and security considerations.

Bridging API and Linked Data

- REST API Linked Data Keywords [github] [I-D] provides a simple way to add semantic information to JSON Schema and OpenAPI documents
- Frictionless Data & Linked Data #218 investigates about providing semantic information to describe Frictionless Data packages containing JSON and CSV files

Follow us

https://innovazione.gov.it/

@InnovazioneGov

@DipartimentoTrasformazioneDigitale

@company/ministeroinnovazione/

DIPARTIMENTO
PER LA TRASFORMAZIONE
DIGITALE