FAETERJ-Rio

Faculdade de Educação Tecnológica do Rio de Janeiro

DISCIPLINA: PRODUÇÃO DE SOFTWARE (PSW)

TEMA: API JAVA JENA

Equipe:

- Jorge Antonio Ferreira Dargam

- Rafael Marques de Albuquerque

- Márcio Pacheco de Lima

- Fábio Faraco

- André Leite Marinho

- Darlan

SUMÁRIO

1 - ONTOLOGIA
2 - VOCABULÁRIO (RDF)
3 - SINTAXE (XML)
4 - FRAMEWORK JENA

FONTE DE CONSULTA

https://jena.apache.org/

Apache Jena

A free and open source Java framework for building Semantic Web and Linked Data applications.

Get started now! | Download

RDF

RDF API

Interact with the core API to create and read Resource Description Framework (RDF) graphs. Serialise your triples using popular formats such as RDF/XML or Turtle.

ARQ (SPARQL)

Query your RDF data using ARQ, a SPARQL 1.1 compliant engine. ARQ supports remote federated queries and free text search.

Triple store

TDB

Persist your data using TDB, a native high performance triple store. TDB supports the full range of Jena APIs.

Fuseki

Expose your triples as a SPARQL end-point accessible over HTTP. Fuseki provides REST-style interaction with your RDF data.

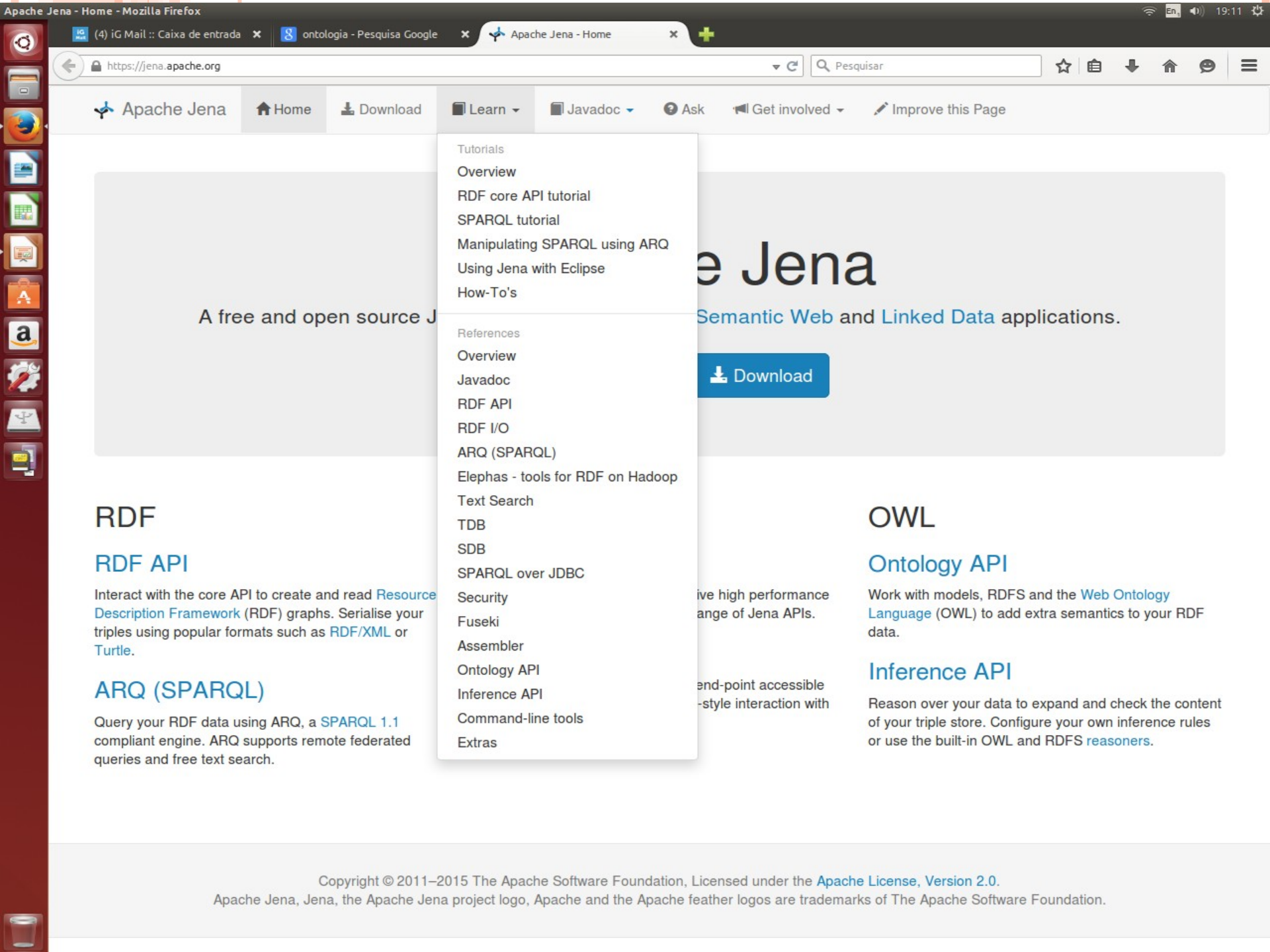
OWL

Ontology API

Work with models, RDFS and the Web Ontology Language (OWL) to add extra semantics to your RDF data.

Inference API

Reason over your data to expand and check the content of your triple store. Configure your own inference rules or use the built-in OWL and RDFS reasoners.


Apache Jena - Home - Mozilla Firefox
(4) iG Mail :: Caixa de entrada
ontologia - Pesquisa Google
Apache Jena - Home
https://jena.apache.org
Pesquisar
Apache Jena
Home
Download
Learn
Javadoc
Ask
Get involved
Improve this Page
Tutorials
Overview
RDF core API tutorial
SPARQL tutorial
Manipulating SPARQL using ARQ
Using Jena with Eclipse
How-To's
References
Overview
Javadoc
RDF API
RDF I/O
ARQ (SPARQL)
Elephas - tools for RDF on Hadoop
Text Search
TDB
SDB
SPARQL over JDBC
Security
Fuseki
Assembler
Ontology API
Inference API
Command-line tools
Extras
e Jena
A free and open source J
Semantic Web and Linked Data applications.
Download
RDF
RDF API
Interact with the core API to create and read Resource Description Framework (RDF) graphs. Serialise your triples using popular formats such as RDF/XML or Turtle.
ARQ (SPARQL)
Query your RDF data using ARQ, a SPARQL 1.1 compliant engine. ARQ supports remote federated queries and free text search.
ive high performance
ange of Jena APIs.
end-point accessible
-style interaction with
OWL
Ontology API
Work with models, RDFS and the Web Ontology Language (OWL) to add extra semantics to your RDF data.
Inference API
Reason over your data to expand and check the content of your triple store. Configure your own inference rules or use the built-in OWL and RDFS reasoners.
Copyright © 2011–2015 The Apache Software Foundation, Licensed under the Apache License, Version 2.0.
Apache Jena, Jena, the Apache Jena project logo, Apache and the Apache feather logos are trademarks of The Apache Software Foundation.

TUTORIALS

Jena tutorials

The following tutorials take a step-by-step approach to explaining aspects of RDF and linked-data applications programming in Jena. For a more task-oriented description, please see the getting started guide.

- RDF core API tutorial
- SPARQL tutorial
- Using Jena with Eclipse
- Manipulating SPARQL using ARQ

Jena tutorials in other languages

Quelques uns des tutoriels de Jena sont aussi disponibles en français. Vous pouvez les voir en suivant ces liens:

- Une introduction à RDF
- Requêtes SPARQL utilisant l'API Java ARQ
- Les entrées/sorties RDF
- Une introduction à SPARQL

Os tutoriais a seguir explicam aspectos de RDF e da programação em Jena de aplicações linked-data. Veja também o guia getting started - em inglês.

- Uma introdução à API RDF
- Tutorial SPARQL
- Manipulando SPARQL usando ARQ
- Usando o Jena com o Eclipse

Traditional Chinese:

- RDF 和 Jena RDF API 入门

1- ONTOLOGIA

Avançar direto para a implementação (JENA), sem conhecer inicialmente o **modelo de dados de RDF**, levará à frustração e ao desapontamento. No entanto, estudar unicamente o modelo de dados é desgastante e muitas vezes leva a "enigmas metafísicos torturantes".

Fonte: https://jena.apache.org/tutorials/rdf_api_pt.html

1- ONTOLOGIA

Filosofia - Ontologia (do grego ontos "ente" e logoi, "ciência do ser") é a parte da metafísica que trata da natureza, realidade e existência dos entes. A ontologia trata do ser enquanto ser, isto é, do ser concebido como tendo uma natureza comum que é inerente a todos e a cada um dos seres.

Fonte: http://pt.wikipedia.org/wiki/Ontologia

1- ONTOLOGIA

Em Ciência da Computação, Sistemas de Informação e Ciência da Informação, uma **ontologia** é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Uma ontologia é utilizada para realizar inferência sobre os objetos do domínio.

Fonte: http://pt.wikipedia.org/wiki/Ontologia_(ciência_da_computação)

1- ONTOLOGIA

Ontologias são utilizadas em inteligência artificial, web semântica, engenharia de software e arquitetura da informação, como uma forma de **representação de conhecimento** sobre o mundo ou alguma parte deste.

Fonte: http://pt.wikipedia.org/wiki/Ontologia_(ciência_da_computação)

1- ONTOLOGIA

Ontologias geralmente descrevem:

- **Indivíduos**: os objetos básicos;
- Classes: conjuntos, coleções ou tipos de objetos;
- Atributos: **propriedades**, características ou parâmetros que os objetos podem ter e compartilhar;
- **Relacionamentos**: formas como os objetos podem se relacionar com outros objetos.

Fonte: http://pt.wikipedia.org/wiki/Ontologia_(ciência_da_computação)

1- ONTOLOGIA

Ontologias são compostas de :

- **vocabulário**: palavras que representam os objetos;

- **sintaxe**: organização da representação das ideias;

- **metadados**: “dados que representam dados”, organização estruturada de vocabulário e sintaxe para representar um dado.

2 – VOCABULÁRIO

objeto

nome

MOCHILA

Vocabulário
(palavra que
representa o objeto)

INDIVÍDUO → PROPRIEDADE → VALOR

2 – VOCABULÁRIO

nome → MOCHILA

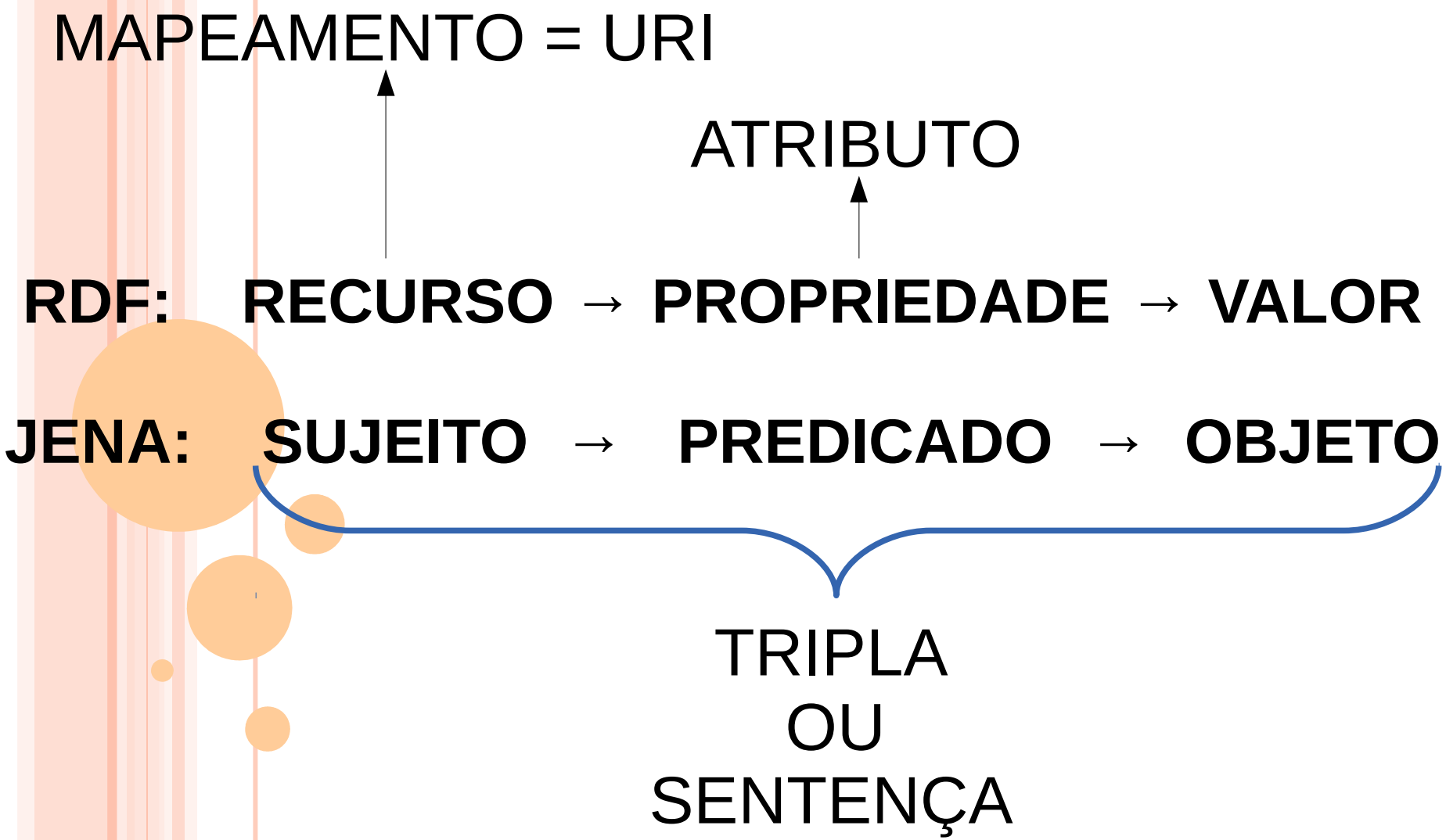
objeto

sintaxe

RDF: **RECURSO → PROPRIEDADE → VALOR**

JENA: **SUJEITO → PREDICADO → OBJETO**

2 – VOCABULÁRIO (RDF)



2 – VOCABULÁRIO (RDF)

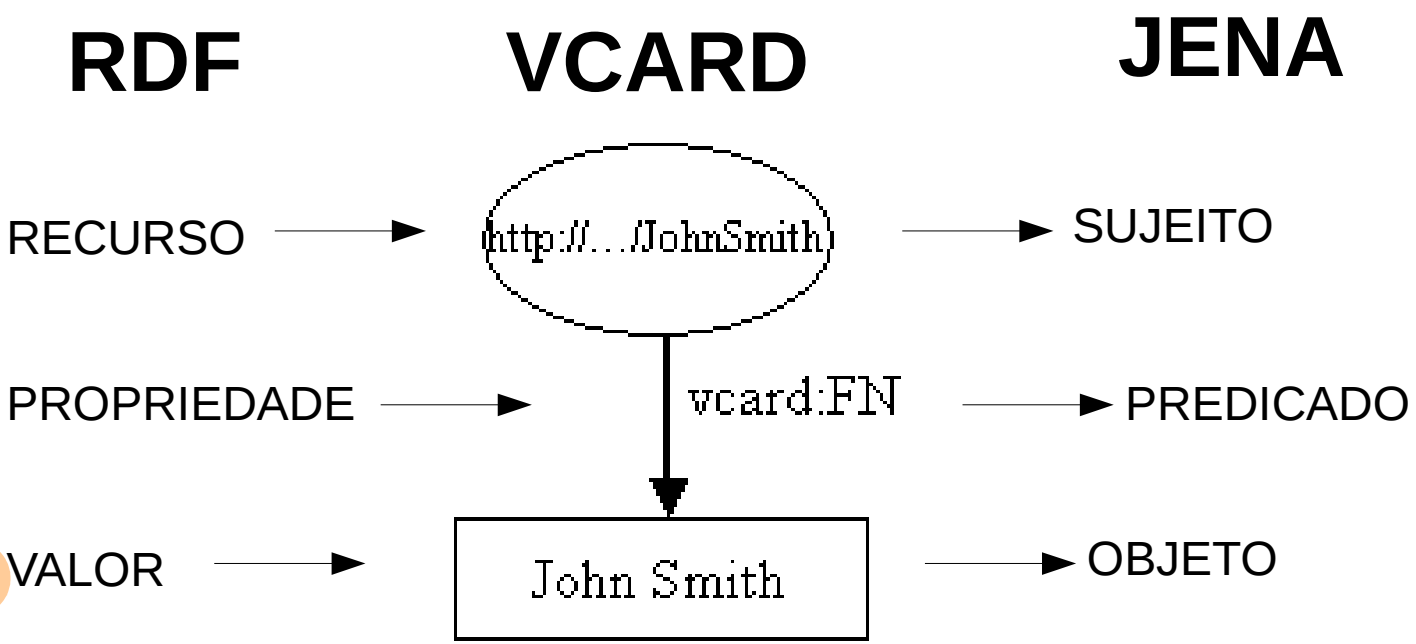
URI (Universal Resource Identificator)

Localizador de recursos

URL (Universal Resource Locator) e URN Universal Resource Name são URI's

Exemplos:
ftp://ftp.is.co.za/rfc/rfc1808.txt
mailto:andrew@linux.ime.usp.br

2 – VOCABULÁRIO (RDF)



Fonte: https://jena.apache.org/tutorials/rdf_api_pt.html

3 – SINTAXE (XML)

O modelo RDF utiliza o XML para representar sua sintaxe.

RDF deve ser entendido em termos do seu **modelo de dados**.

Os dados RDF podem ser representados (escritos) em XML.

Fonte: https://jena.apache.org/tutorials/rdf_api_pt.html

3 – SINTAXE (XML)

Existem outros modelos:

- RDF
- SHOE
- XOL
- OML
- DAML
- OIL
- OWL

Fonte: Ontologias e a Biblioteca Virtual – Andrew Gan King Yuan

3 – SINTAXE (XML)

Existem outros modelos:

- RDF
- SHOE
- XOL
- OML
- DAML
- OIL
- OWL

Mais usados

Fonte: Ontologias e a Biblioteca Virtual – Andrew Gan King Yuan

3 – SINTAXE (XML)

O framework de descrição de recursos (RDF) é um padrão (tecnicamente uma recomendação da W3C) para descrever **recursos**.
Os recursos usam uma representação RDF de cartão de negócios (VCARDS). RDF é melhor representado como um diagrama de nós e arcos.

Fonte: https://jena.apache.org/tutorials/rdf_api_pt.html

3 – SINTAXE (XML)

Indica que este não é um recurso “em branco”

RECURSO

CLASSE

URI

```
<vcard:Individual rdf:about="http://example.com/me/corky">
```

VALOR

```
 <vcard:fn>Corky Crystal</vcard:fn>
```

PROPRIEDADE

```
 <vcard:nickname>Corks</vcard:nickname>

 <vcard:hasEmail rdf:resource="mailto:corky@example.com"/>
```

Esta propriedade é um recurso (N-Ary Property)

```
</vcard:Individual>
```

Fonte: http://www.w3.org/TR/vcard-rdf/

3 – SINTAXE (XML)

```
<vcard:Individual rdf:about="http://example.com/me/corky">
    <vcard:fn>Corky Crystal</vcard:fn>
        <vcard:hasNickname rdf:parseType="Resource">
            <vcard:value>Corks</vcard:value>
            <vcard:sort-string>cork</vcard:sort-string>
        </vcard:hasNickname>
 <vcard:hasEmail rdf:parseType="Resource">
  <vcard:hasValue rdf:resource="mailto:corky@example.com"/>
  <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Home"/>
 </vcard:hasEmail>
</vcard:Individual>
```

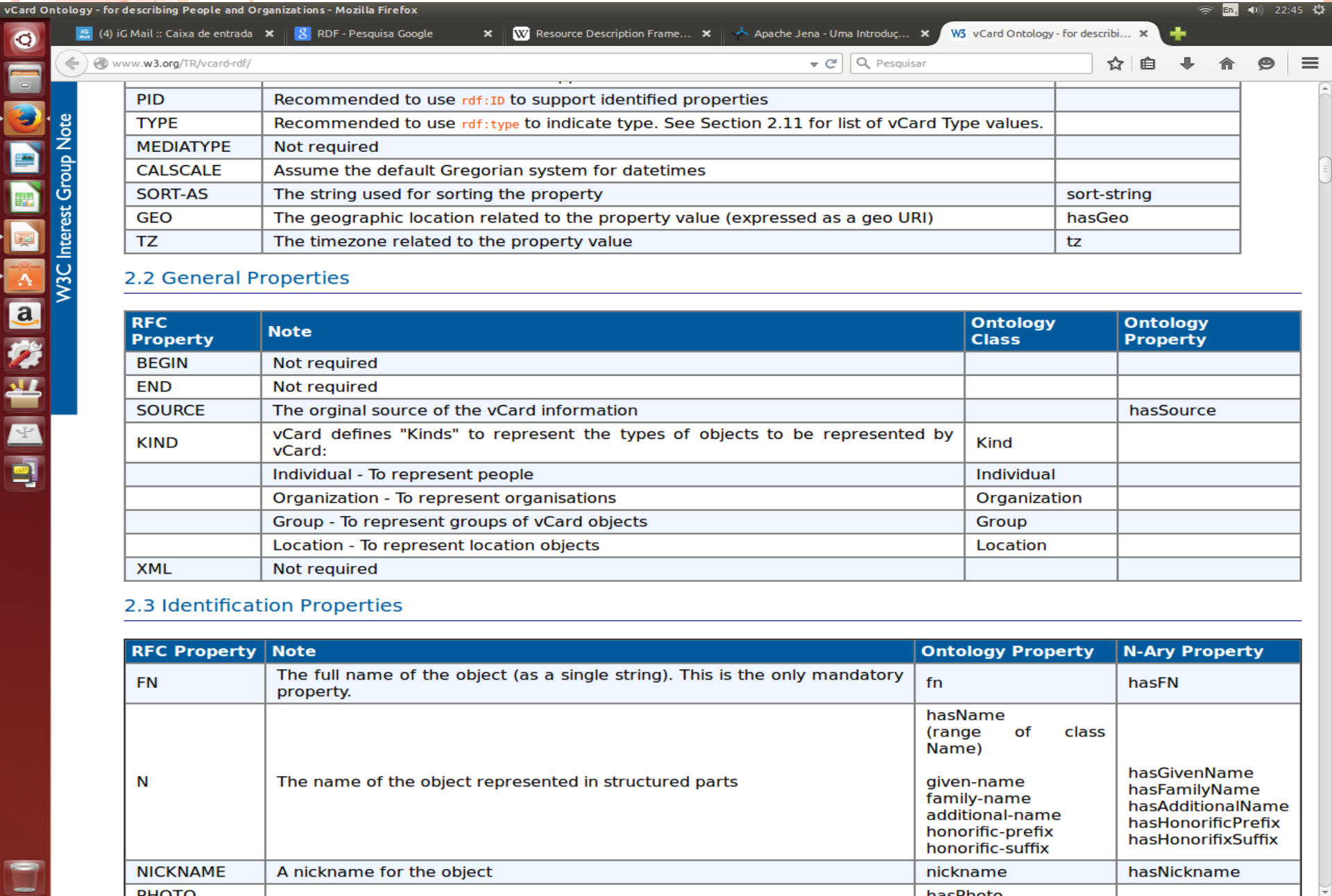
Fonte: http://www.w3.org/TR/vcard-rdf/

3 – SINTAXE (XML)

```
<vcard:Individual rdf:about="http://example.com/me/corky">
    <vcard:fn>Corky Crystal</vcard:fn>
        <vcard:hasNickname rdf:parseType="Resource">
            <vcard:value>Corks</vcard:value>
            <vcard:sort-string>cork</vcard:sort-string>
        </vcard:hasNickname>
 <vcard:hasEmail rdf:parseType="Resource">
  <vcard:hasValue rdf:resource="mailto:corky@example.com"/>
  <rdf:type rdf:resource="http://www.w3.org/2006/vcard/ns#Home"/>
 </vcard:hasEmail>
</vcard:Individual>
```

Fonte: http://www.w3.org/TR/vcard-rdf/

3 – SINTAXE (XML)



PID	Recommended to use `rdf:ID` to support identified properties	
TYPE	Recommended to use `rdf:type` to indicate type. See Section 2.11 for list of vCard Type values.	
MEDIATYPE	Not required	
CALSCALE	Assume the default Gregorian system for datetimes	
SORT-AS	The string used for sorting the property	sort-string
GEO	The geographic location related to the property value (expressed as a geo URI)	hasGeo
TZ	The timezone related to the property value	tz

2.2 General Properties

RFC Property	Note	Ontology Class	Ontology Property
BEGIN	Not required		
END	Not required		
SOURCE	The orginal source of the vCard information		hasSource
KIND	vCard defines "Kinds" to represent the types of objects to be represented by vCard:	Kind	
	Individual - To represent people	Individual	
	Organization - To represent organisations	Organization	
	Group - To represent groups of vCard objects	Group	
	Location - To represent location objects	Location	
XML	Not required		

2.3 Identification Properties

RFC Property	Note	Ontology Property	N-Ary Property
FN	The full name of the object (as a single string). This is the only mandatory property.	fn	hasFN
N	The name of the object represented in structured parts	hasName (range of class Name) given-name family-name additional-name honorific-prefix honorific-suffix	 hasGivenName hasFamilyName hasAdditionalName hasHonorificPrefix hasHonorifixSuffix
NICKNAME	A nickname for the object	nickname	hasNickname
PHOTO		hasPhoto	

3 – SINTAXE (XML)

vCard Ontology - for describing People and Organizations - Mozilla Firefox

(4) iG Mail :: Caixa de entrada | RDF - Pesquisa Google | Resource Description Frame... | Apache Jena - Uma Introduç... | vCard Ontology - for describi...

www.w3.org/TR/vcard-rdf/

W3C Interest Group Note

RFC Property	Note	Ontology Class	Ontology Property
BEGIN	Not required		
END	Not required		
SOURCE	The orginal source of the vCard information		hasSource
KIND	vCard defines "Kinds" to represent the types of objects to be represented by vCard:	Kind	
	Individual - To represent people	Individual	
	Organization - To represent organisations	Organization	
	Group - To represent groups of vCard objects	Group	
	Location - To represent location objects	Location	
XML	Not required		

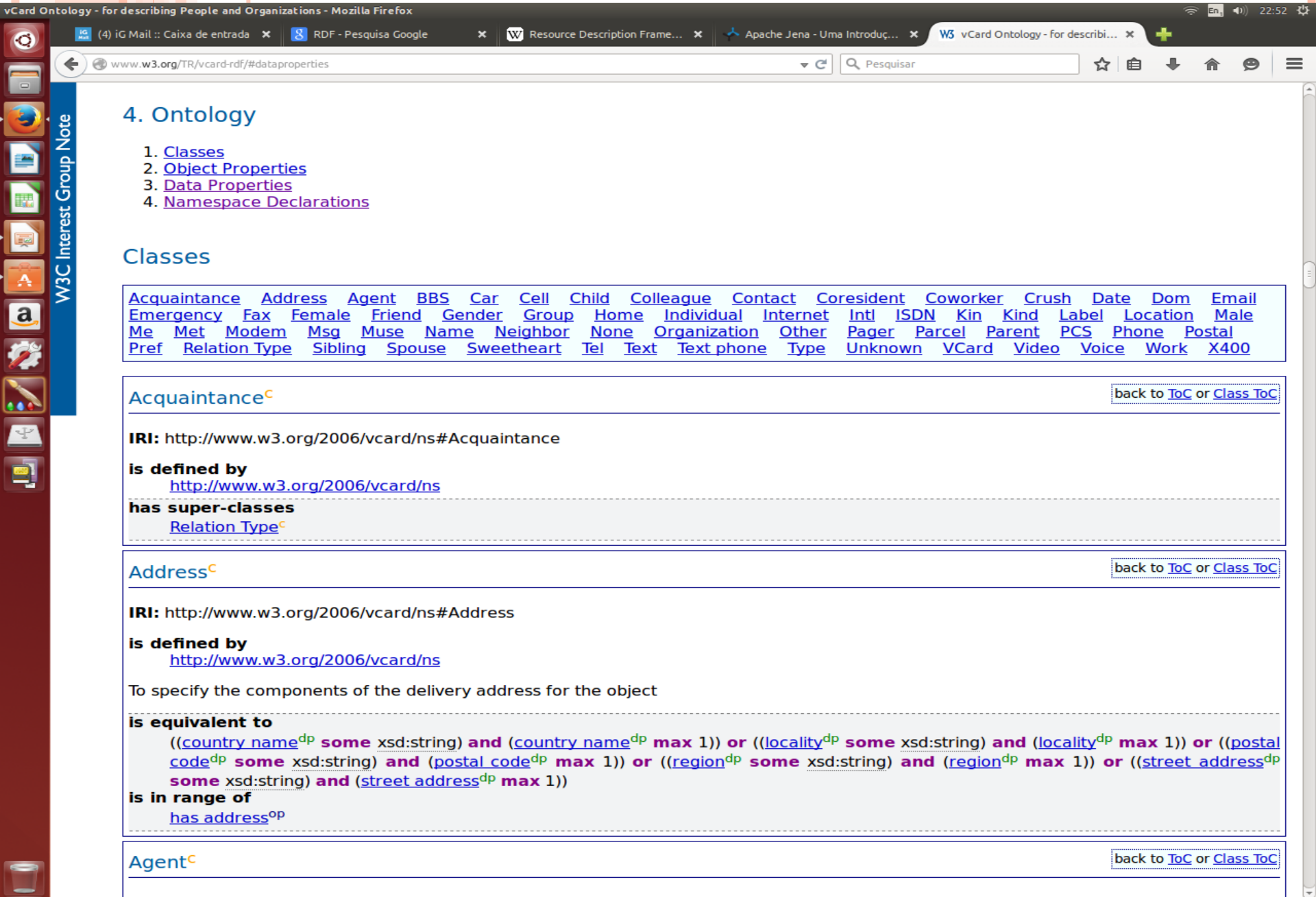
2.3 Identification Properties

RFC Property	Note	Ontology Property	N-Ary Property
FN	The full name of the object (as a single string). This is the only mandatory property.	fn	hasFN
N	The name of the object represented in structured parts	hasName (range of class Name) given-name family-name additional-name honorific-prefix honorific-suffix	 hasGivenName hasFamilyName hasAdditionalName hasHonorificPrefix hasHonorifixSuffix
NICKNAME	A nickname for the object	nickname	hasNickname
PHOTO		hasPhoto	
BDAY	Birth date of the object. Should only apply to Individual.	bday	
ANNIVERSARY	Should only apply to Individual	anniversary	
GENDER	Should only apply to Individual. See Gender Codes in Section 2.11.	hasGender	

2.4 Delivery Addressing Properties

RFC Property	Note	Ontology Property	N-Ary Property
		hasAddress (range of class Address)	

3 – SINTAXE (XML)



4. Ontology

1. Classes
2. Object Properties
3. Data Properties
4. Namespace Declarations

Classes

Acquaintance Address Agent BBS Car Cell Child Colleague Contact Coresident Coworker Crush Date Dom Email Emergency Fax Female Friend Gender Group Home Individual Internet Intl ISDN Kin Kind Label Location Male Me Met Modem Msg Muse Name Neighbor None Organization Other Pager Parcel Parent PCS Phone Postal Pref Relation Type Sibling Spouse Sweetheart Tel Text Text phone Type Unknown VCard Video Voice Work X400

Acquaintance[c]

back to ToC or Class ToC

IRI: http://www.w3.org/2006/vcard/ns#Acquaintance

is defined by
http://www.w3.org/2006/vcard/ns

has super-classes
Relation Type[c]

Address[c]

back to ToC or Class ToC

IRI: http://www.w3.org/2006/vcard/ns#Address

is defined by
http://www.w3.org/2006/vcard/ns

To specify the components of the delivery address for the object

is equivalent to
((country name[dp] **some** xsd:string) **and** (country name[dp] **max** 1)) **or** ((locality[dp] **some** xsd:string) **and** (locality[dp] **max** 1)) **or** ((postal code[dp] **some** xsd:string) **and** (postal code[dp] **max** 1)) **or** ((region[dp] **some** xsd:string) **and** (region[dp] **max** 1)) **or** ((street address[dp] **some** xsd:string) **and** (street address[dp] **max** 1))

is in range of
has address[op]

Agent[c]

back to ToC or Class ToC

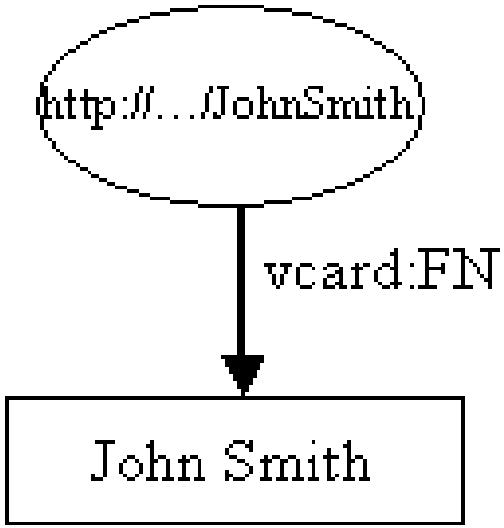
4 – FRAMEWORK JENA

Jena é uma API Java que pode ser usada para pra criar e manipular grafos RDF.
Jena possui classes para representar grafos, recursos, propriedades e literais.
As interfaces que representam recursos, propriedades e literais são chamadas de modelo e é representada pela interface Model.

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

4 – FRAMEWORK JENA

```
<vcard:Individual rdf:about="http://somewhere/JohnSmith">
    <vcard:fn>John Smith</vcard:fn>
</vcard:Individual>
```



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

4 – FRAMEWORK JENA

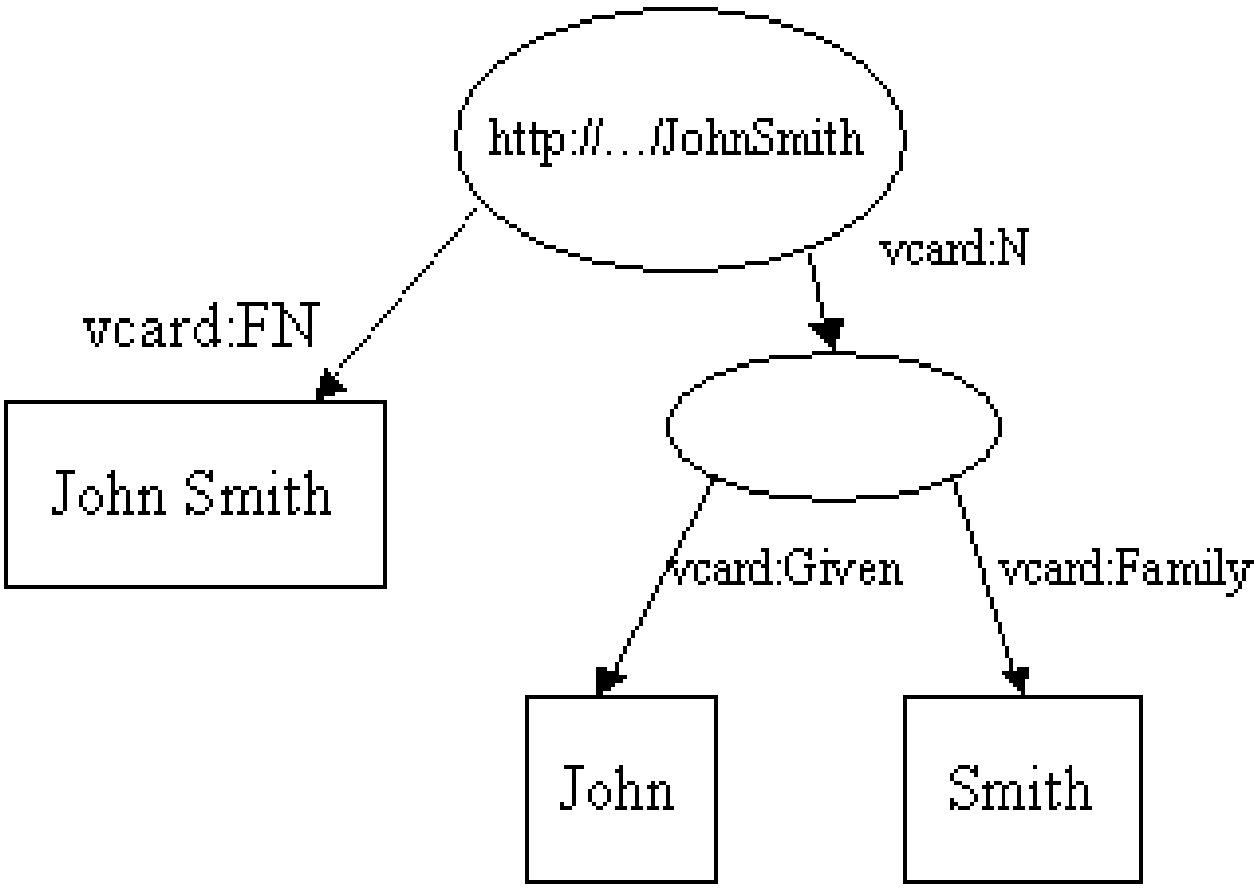
```
// definições
static String personURI = "http://somewhere/JohnSmith";
static String fullName = "John Smith";

// cria um modelo de RDF vazio
Model model = ModelFactory.createDefaultModel();

// cria um recurso
Resource johnSmith =
model.createResource(personURI);

// adiciona uma propriedade ao recurso
 johnSmith.addProperty(VCARD.FN, fullName);
```

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

4 – FRAMEWORK JENA

```
package jena.examples.rdf ;

import com.hp.hpl.jena.rdf.model.*;
import com.hp.hpl.jena.vocabulary.*;

/** Tutorial 1 creating a simple model
 */

public class Tutorial01 extends Object {
    // some definitions
    static String personURI    = "http://somewhere/JohnSmith";
    static String fullName     = "John Smith";

      public static void main (String args[]) {
        // create an empty model
        Model model = ModelFactory.createDefaultModel();

       // create the resource
       Resource johnSmith = model.createResource(personURI);

      // add the property
      johnSmith.addProperty(VCARD.FN, fullName);
      }
}
```

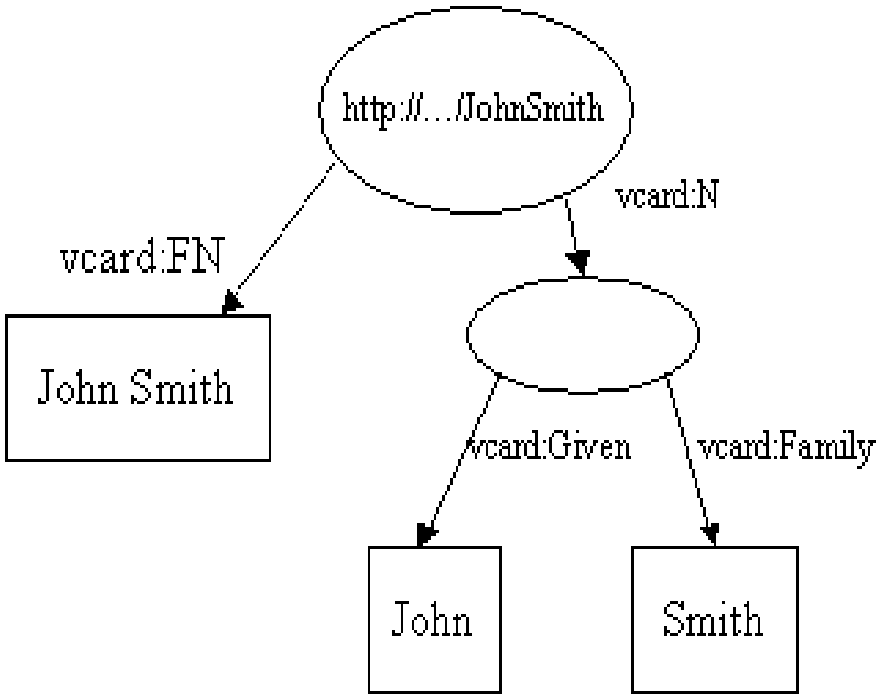
4 – FRAMEWORK JENA



Obs.:
Blank node
(nó em branco)

4 – FRAMEWORK JENA

```
<vcard:Individual rdf:about="http://somewhere/JohnSmith">
      <vcard:fn>John Smith</vcard:fn>
      <vcard:N>
            <vcard:Given>John</vcard:Given>
            <vcard:Family>Smith</vcard:Family>
      </vcard:N>
</vcard:Individual>
```



4 – FRAMEWORK JENA

```
// some definitions
String personURI    = "http://somewhere/JohnSmith";
String givenName    = "John";
String familyName   = "Smith";
String fullName     = givenName + " " + familyName;

// create an empty Model
Model model = ModelFactory.createDefaultModel();

// create the resource
//   and add the properties cascading style
Resource johnSmith
  = model.createResource(personURI)
         .addProperty(VCARD.FN, fullName)
         .addProperty(VCARD.N,
                      model.createResource()
                           .addProperty(VCARD.Given, givenName)
                           .addProperty(VCARD.Family, familyName));
```

4 – FRAMEWORK JENA

```
public class Tutorial02 extends Object {

    public static void main (String args[]) {
      // some definitions
      String personURI    = "http://somewhere/JohnSmith";
      String givenName    = "John";
      String familyName   = "Smith";
      String fullName     = givenName + " " + familyName;

      // create an empty model
      Model model = ModelFactory.createDefaultModel();

      // create the resource
      //   and add the properties cascading style
      Resource johnSmith  = model.createResource(personURI)
            .addProperty(VCARD.FN, fullName)
            .addProperty(VCARD.N,
                    model.createResource()
                          .addProperty(VCARD.Given, givenName)
                          .addProperty(VCARD.Family, familyName));

    }
}
```

4 – FRAMEWORK JENA

O Jena lida com nós em branco criando identificadores internos estabelecendo um link entre o recurso e as propriedades.

anon:14df86:ecc3dee17b:-7fff

```
http://somewhere/JohnSmith http://www.w3.org/2001/vcard-rdf/3.0#N anon:14df86:ecc3dee17
b:-7fff .
anon:14df86:ecc3dee17b:-7fff http://www.w3.org/2001/vcard-rdf/3.0#Family  "Smith" .
anon:14df86:ecc3dee17b:-7fff http://www.w3.org/2001/vcard-rdf/3.0#Given  "John" .
http://somewhere/JohnSmith http://www.w3.org/2001/vcard-rdf/3.0#FN  "John Smith" .
```

4 – FRAMEWORK JENA

Escrita de RDF

Jena possui métodos para ler e escrever RDF como XML. Eles podem ser usados para armazenar o modelo RDF em um arquivo e carregá-lo novamente em outro momento.

```
// write the model in XML form to a file
model.write(System.out);
```

4 – FRAMEWORK JENA

```
<rdf:RDF
 xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
 xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
>
 <rdf:Description rdf:about='http://somewhere/JohnSmith'>
   <vcard:FN>John Smith</vcard:FN>
   <vcard:N rdf:nodeID="A0"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A0">
   <vcard:Given>John</vcard:Given>
   <vcard:Family>Smith</vcard:Family>
 </rdf:Description>
</rdf:RDF>
```

4 – FRAMEWORK JENA

```
<rdf:RDF
 xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
 xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
>
 <rdf:Description rdf:about='http://somewhere/JohnSmith'>
   <vcard:FN>John Smith</vcard:FN>
   <vcard:N rdf:nodeID="A0"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A0">
   <vcard:Given>John</vcard:Given>
   <vcard:Family>Smith</vcard:Family>
 </rdf:Description>
</rdf:RDF>
```

ERRO
Deixou de ser
Blanck node

4 – FRAMEWORK JENA

```
<rdf:RDF
 xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
 xmlns:vcard='http://www.w3.org/2001/vcard-rdf/3.0#'
>
 <rdf:Description rdf:about='http://somewhere/JohnSmith'>
   <vcard:FN>John Smith</vcard:FN>
   <vcard:N rdf:nodeID="A0"/>
 </rdf:Description>
 <rdf:Description rdf:nodeID="A0">
   <vcard:Given>John</vcard:Given>
   <vcard:Family>Smith</vcard:Family>
 </rdf:Description>
</rdf:RDF>
```

ERRO
Deixou de ser
Blanck node

REFERÊNCIAS NO DECORRER DA APRESENTAÇÃO