

Tema 8: La Web Semántica

Modelando la semántica

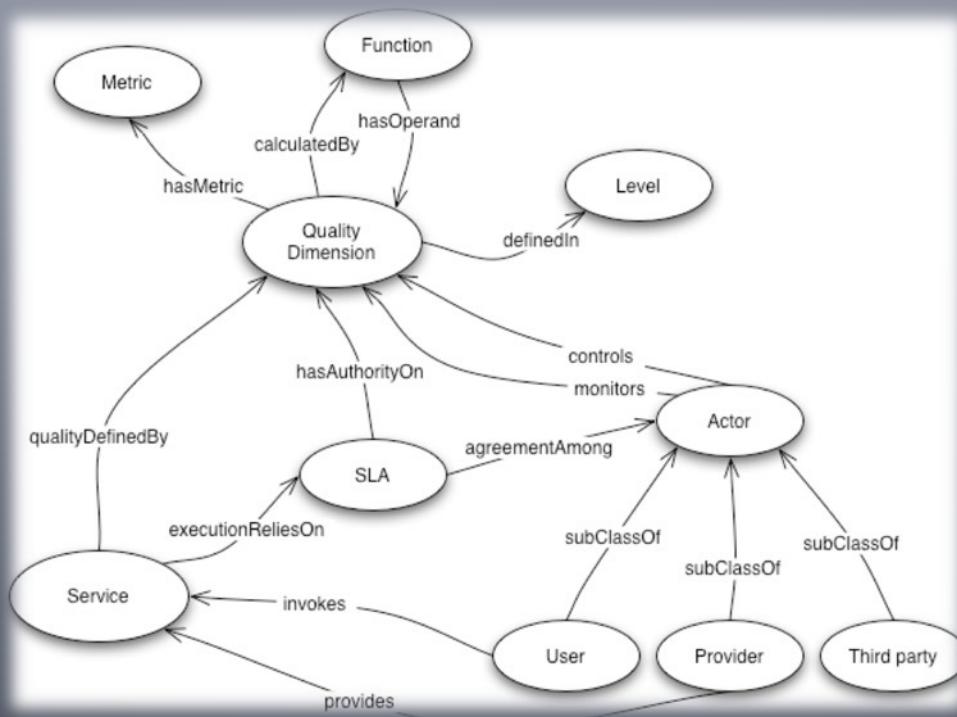
RDF y RDFa

SPARQL

Sistemas de Información para Internet
3º del Grado de Ingeniería Informática (tres menciones)

Departamento de Lenguajes y Ciencias de la Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Modelando la semántica



Modelando la semántica

Motivación

- La Web actual contiene mucha información
- Pero la mayoría de la información está pensada para ser consumida por humanos
- Hasta hace poco las búsquedas eran sintácticas, se buscan páginas que contengan una determinada palabra o frase
 - Ej.: busca páginas que contengan “Java EE 7”
- Hoy en día la Web está cambiando
 - Ej.: busca imágenes de color rojo (en google)

Modelando la semántica

Motivación

- Pero aún queda mucho por hacer
 - Ej.: busca lista de planetas enanos
 - Ej.: busca actores/rices que compartieron reparto con James Dean
- La información no suele estar organizada de una forma que las máquinas puedan entender
 - Ej.: No pueden saber si el “James Dean” que aparece en las distintas páginas son el mismo “James Dean” (¿y nosotros podemos? ¿cómo?)

Modelando la semántica Informal y formal

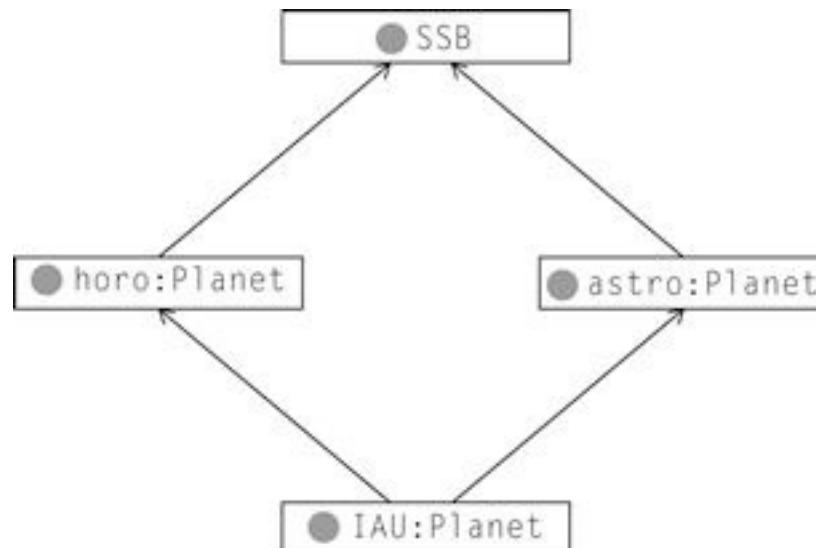
- Si queremos que las máquinas entiendan la información y puedan sacar conclusiones debemos modelar la semántica
- Es posible usar **modelos informales**: descripciones textuales indicando lo que significa cada cosa
 - Los usamos diariamente
- Pero de nuevo, esto solo lo entienden los humanos
 - En la Web también se hace: folksonomía (etiquetado)

Modelando la semántica Informal y formal

- Pero si queremos que las máquinas realmente puedan procesar la semántica (igual que hacen con la sintaxis) necesitamos modelos formales
 - Todo tiene una definición precisa y basada en la “forma” de las entidades
 - Existen reglas de manipulación precisamente definidas
 - Ej.: las matemáticas se usan para modelar el mundo, la lógica se usa para modelar conocimiento
 - Los modelos permiten explicar sucesos y predecir otros (e.g., el próximo eclipse, el bosón de Higgs, ...)

Modelando la semántica Informal y formal

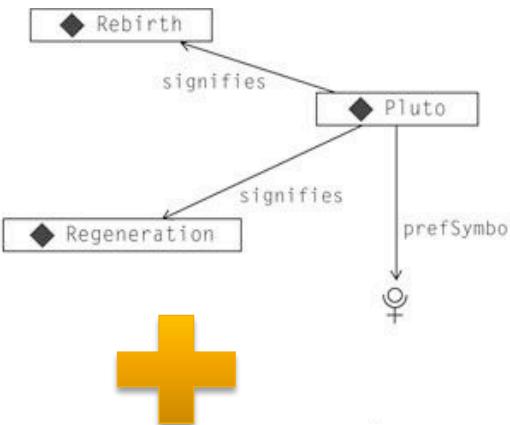
- Este modelo debe permitir variabilidad
 - Los conceptos no tienen las mismas características para todos los sujetos
 - Ej.: ¿Qué es un planeta?



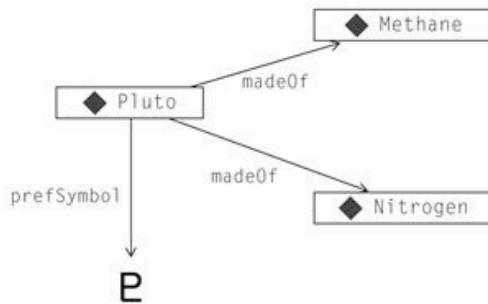
Modelando la semántica Informal y formal

- Debe permitir unir información procedente de varias fuentes

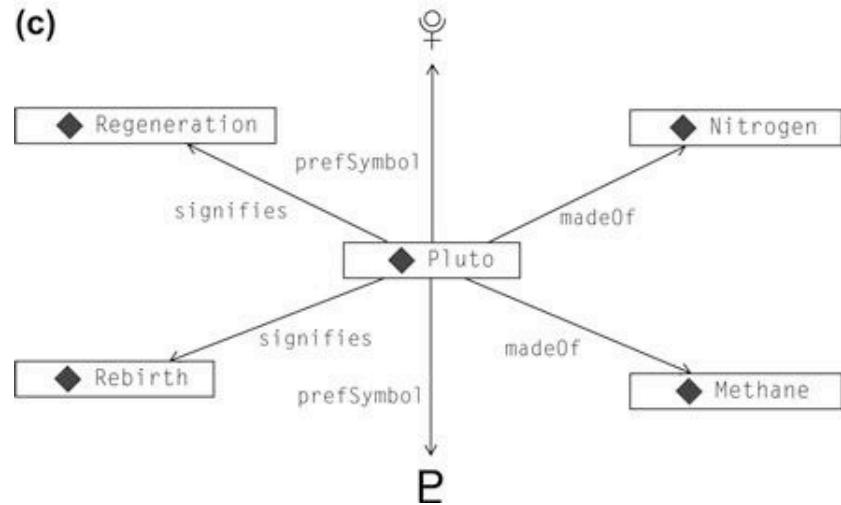
(a)



(b)



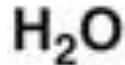
(c)



Modelando la semántica Informal y formal

- Debe permitir una descripción de las entidades con distinto nivel de detalle

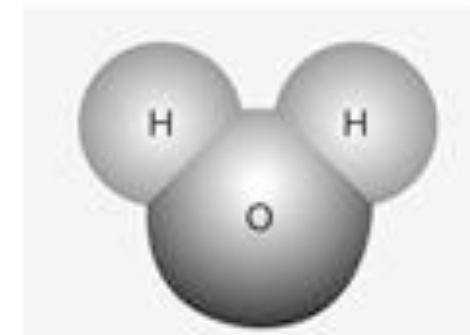
(a)



(b)



(c)



- Debe permitir “a cada uno decir lo que quiera de cualquier cosa” (AAA slogan: *Anyone can say Anything about Any topic*)

Modelando la semántica

Lenguajes de modelado semántico

- Los lenguajes utilizados para el modelado semántico en la Web (del menos expresivo al más expresivo) son:
 - RDF (*Resource Description Framework*)
 - RDFS (*RDF Schema*)
 - RDFS-Plus
 - OWL (*Web Ontology Language*)

RDF y RDFa



RDF y RDFa

Introducción

- RDF es el más básico de los lenguajes para el modelado semántico de la Web y es en el que están basados todos los demás
- RDF se centra en la gestión de datos distribuidos
- En la Web semántica, se denomina **recurso** (*resource*) a los objetos del mundo
- De ahí que RDF sea acrónimo de *Resource Description Framework*

RDF y RDFa

Ternas

- El elemento básico de información en RDF es una **terna** con **sujeto**, **predicado** (o propiedad) y **objeto**

Table 3.3 Sample Triples

Subject	Predicate	Object
Shakespeare	wrote	King Lear
Shakespeare	wrote	Macbeth
Anne Hathaway	married	Shakespeare
Shakespeare	livedIn	Stratford
Stratford	isIn	England
Macbeth	setIn	Scotland
England	partOf	UK
Scotland	partOf	UK

RDF y RDFa

Ternas

- Un conjunto de ternas puede formar un grafo (que recuerda a un grafo conceptual)

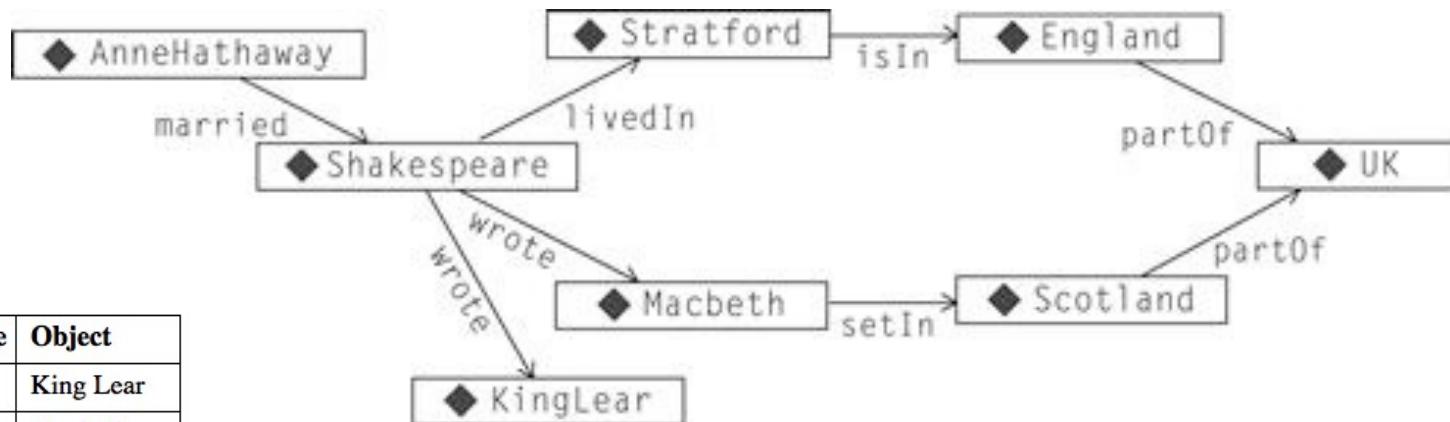


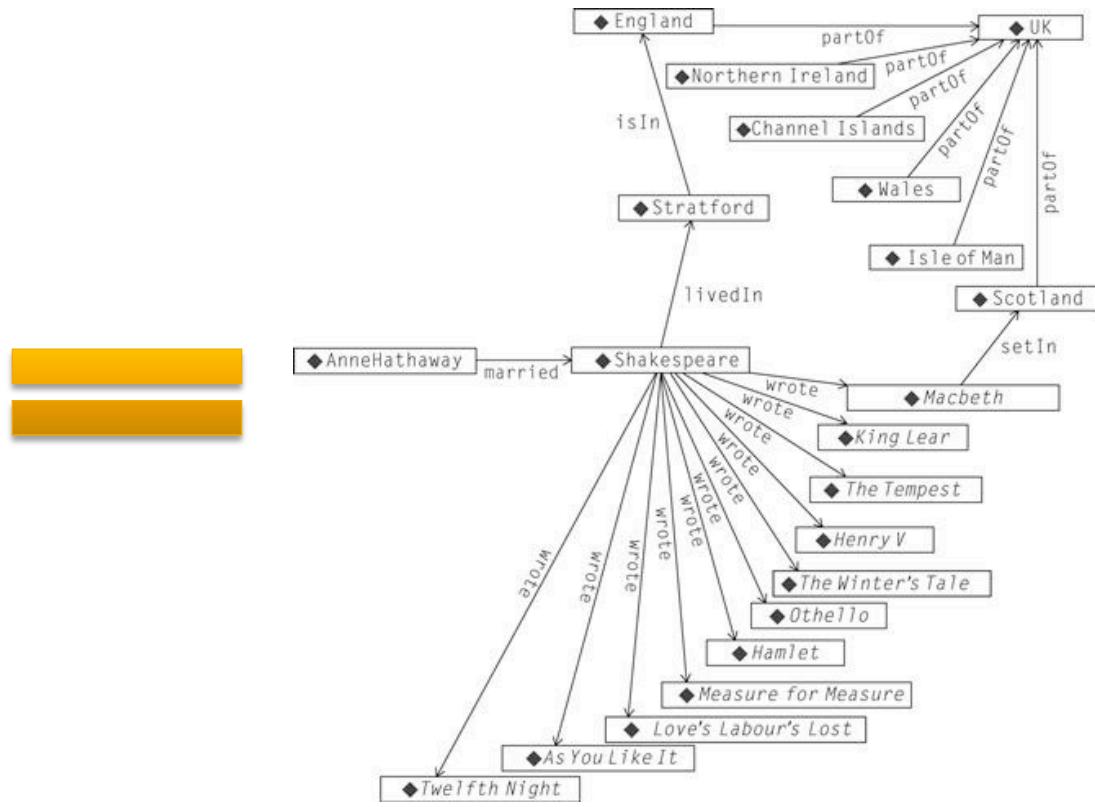
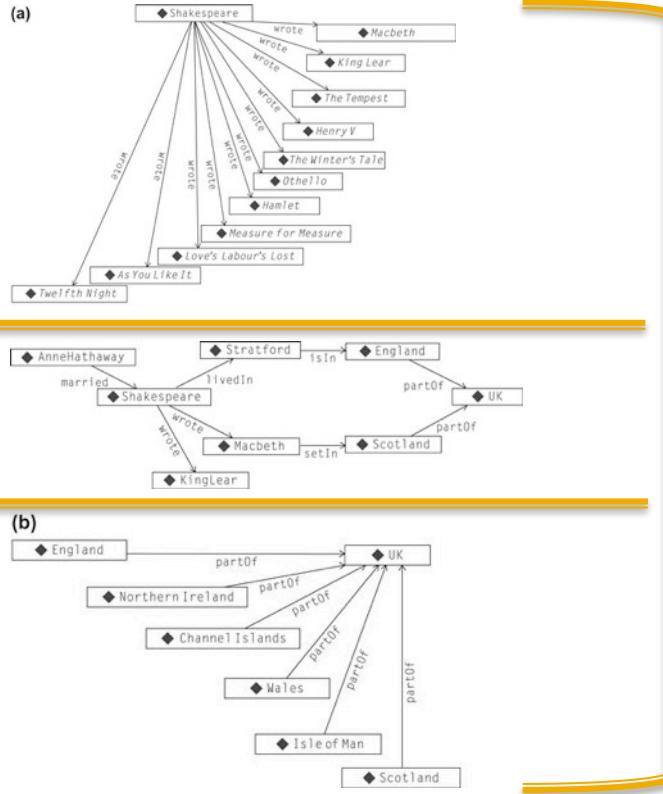
Table 3.3 Sample Triples

Subject	Predicate	Object
Shakespeare	wrote	King Lear
Shakespeare	wrote	Macbeth
Anne Hathaway	married	Shakespeare
Shakespeare	livedIn	Stratford
Stratford	isIn	England
Macbeth	setIn	Scotland
England	partOf	UK
Scotland	partOf	UK

RDF y RDFa

Ternas

- Esta representación de la información permite combinar distintas fuentes muy fácilmente



RDF y RDFa

URIs

- ¿Cómo podemos asignar a un objeto del mundo un identificador único en el mundo?
- Asignándole un **URI** (*Universal Resource Identifier*)
- Cuidado!! Un URI no es un URL (no tiene por qué estar enlazado a un documento)
 - Pero si lo está no pasa nada (y puede ofrecerse información adicional sobre el objeto en cuestión)
 - Ej.: <http://data.linkedmdb.org/resource/film/1973>

RDF y RDFa

URIs y qnames

- Los predicados y objetos también se representan mediante URIs
- Para evitar ternas como esta:
 - <<http://data.linkedmdb.org/resource/film/1973>>
<<http://data.linkedmdb.org/resource/movie/actor>>
<<http://data.linkedmdb.org/resource/actor/31386>>
- Se suelen usar **qnames** con un prefijo previamente definido que hace referencia a un espacio de nombres
 - Hemos usado muchos **qnames** en JSF y JSP

RDF y RDFa

URIs y qnames

- Si definimos
 - movie = <<http://data.linkedmdb.org/resource/movie>>
 - film = <<http://data.linkedmdb.org/resource/film>>
 - actor= <<http://data.linkedmdb.org/resource/actor>>
- Podemos escribir la terna como:
 - film:1973 movie:actor actor:31386

Table 3.8 Triples Referring to URIs with a Variety of Namespaces

Subject	Predicate	Object
lit:Shakespeare	lit:wrote	lit:KingLear
lit:Shakespeare	lit:wrote	lit:MacBeth
bio:AnneHathaway	bio:married	lit:Shakespeare
bio:AnneHathaway	bio:livedWith	lit:Shakespeare
lit:Shakespeare	bio:livedIn	geo:Stratford

RDF y RDFa

URLs y qnames

- Existen algunos espacios de nombres y “objetos” definidos ya por RDF y otros lenguajes de modelado semántico
 - `rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#`
 - `rdfs: http://www.w3.org/2000/01/rdf-schema#`
 - `owl: http://www.w3.org/2002/07/owl#`
- Un predicado muy utilizado definido en el espacio rdf es `rdf:type`
 - Permite decir que un objeto es de cierto tipo

Subject	Predicate	Object
<code>lit:wrote</code>	<code>rdf:type</code>	<code>rdf:Property</code>
<code>geo:partOf</code>	<code>rdf:type</code>	<code>rdf:Property</code>

RDF y RDFa

Orden superior

- No solo es posible “predicar” sobre sujetos, sino que también es posible “predicar” sobre “predicados” (ternas)
- Así en lugar de decir:

```
lit:Shakespeare lit:wrote lit:Hamlet.
```

- Podríamos decir: subject, predicate y object definidos en rdf:

```
q:n1 rdf:subject lit:Shakespeare;  
      rdf:predicate lit:wrote;  
      rdf:object lit:Hamlet.
```

```
web:Wikipedia m:says q:n1 .
```

RDF y RDFa

Representación de ternas

- La representación de la secuencia de ternas usando las URIs se denomina *N-triples*
- Una representación más “amigable” es Turtle

```
@prefix mfg:  
  <http://www.WorkingOntologist.com/Examples/Chapter3/Manufac  
    turing#>  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
  
mfg:Product1 rdf:type mfg:Product .
```

- Turtle añade algo de azúcar sintáctico para simplificar la escritura

RDF y RDFa

Representación de ternas

- Las tuplas que se refieren al mismo sujeto se pueden separar por ;

```
mfg:Product1 rdf:type mfg:Product;
    mfg:Product_Division "Manufacturing support";
    mfg:Product_ID "1";
    mfg:Product_Manufacture_Location "Sacramento";
    mfg:Product_ModelNo "ZX-3";
    mfg:Product_Product_Line "Paper Machine";
    mfg:Product_SKU "FB3524";
    mfg:Product_Available "23" .
mfg:Product2 rdf:type mfg:Product;
    mfg:Product_Division "Manufacturing support";
    mfg:Product_ID "2";
    mfg:Product_Manufacture_Location "Sacramento";
    mfg:Product_ModelNo "ZX-3P";
    mfg:Product_Product_Line "Paper Machine";
    mfg:Product_SKU "KD5243";
    mfg:Product_Available "4" .
```

RDF y RDFa

Representación de ternas

- Las tuplas que se refieren al mismo sujeto y predicado se pueden **separar por ,**

```
lit:Shakespeare b:hasChild b:Susanna, b:Judith, b:Hamnet.
```

- Es equivalente a:

```
lit:Shakespeare b:hasChild b:Susanna.  
lit:Shakespeare b:hasChild b:Judith.  
lit:Shakespeare b:hasChild b:Hamnet.
```

- Añade una abreviatura para rdf:type a

```
mfg:Product1 rdf:type mfg:Product.  
lit:Shakespeare rdf:type lit:Playwright.
```



```
mfg:Product1 a mfg:Product.  
lit:Shakespeare a lit:Playwright.
```

RDF y RDFa

Representación de ternas

- A veces podemos querer predicar sobre un objeto del que no conocemos la identidad

```
lit:Mistress1 rdf:type bio:Woman;  
    bio:LivedIn geo:England.  
lit:Sonnet78 lit:hasInspiration lit:Mistress1.
```

- Podemos añadir en ese caso un **bnode (blank node)**

El objeto de la primera terna es el sujeto de la terna entre corchetes, cuya identidad desconocemos

```
lit:Sonnet78 lit:hasInspiration [a :Woman;  
    bio:livedIn geo:England].
```

RDF y RDFa

Representación de ternas

- Podemos indicar orden usando predicados ya definidos en rdf

```
lit:Shakespeare b:hasChild _:a.  
_:a rdf:first b:Susanna.  
_:a rdf:rest _:b.  
_:b rdf:first b:Judith.  
_:b rdf:rest _:c.  
_:c rdf:rest rdf:nil.  
_:c rdf:first b:Hamnet.
```

¿No os recuerda a las listas de Haskell y Prolog?

- Turtle proporciona azúcar sintáctico para las listas

```
lit:Shakespeare b:hasChild (b:Susanna b:Judith b:Hamnet).
```

RDF y RDFa

RDFa

- RDFa (RDF in *attributes*) es un estándar de W3C para añadir extensiones a HTML y XHTML relacionadas con RDF
- La idea es incluir información semántica en las propias páginas Web
- La información semántica se incorpora a los elementos usando atributos

RDF y RDFa

RDFa

XHTML+RDFa

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
 "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:dc="http://purl.org/dc/elements/1.1/"
 version="XHTML+RDFa 1.0" xml:lang="en">
<head>
  <title>John's Home Page</title>
  <base href="http://example.org/john-d/" />
  <meta property="dc:creator" content="Jonathan Doe" />
  <link rel="foaf:primaryTopic" href="http://example.org/john-d/#me" />
</head>
<body about="http://example.org/john-d/#me">
  <h1>John's Home Page</h1>
  <p>My name is <span property="foaf:nick">John D</span> and I like
    <a href="http://www.neubauten.org/" rel="foaf:interest"
       xml:lang="de">Einstürzende Neubauten</a>.
  </p>
  <p>
    My <span rel="foaf:interest" resource="urn:ISBN:0752820907">favorite
    book is the inspiring <span about="urn:ISBN:0752820907"><cite
      property="dc:title">Weaving the Web</cite> by
      <span property="dc:creator">Tim Berners-Lee</span></span></span>.
  </p>
</body>
</html>
```

The diagram shows an example of an XHTML+RDFa document with annotations:

- Recurso:** Points to the subject of the RDF triple, which is the current element being processed.
- Predicado:** Points to the predicate of the RDF triple, which is the attribute used to map the element to the vocabulary.
- Objeto:** Points to the object of the RDF triple, which is the value of the attribute or the content of the element.

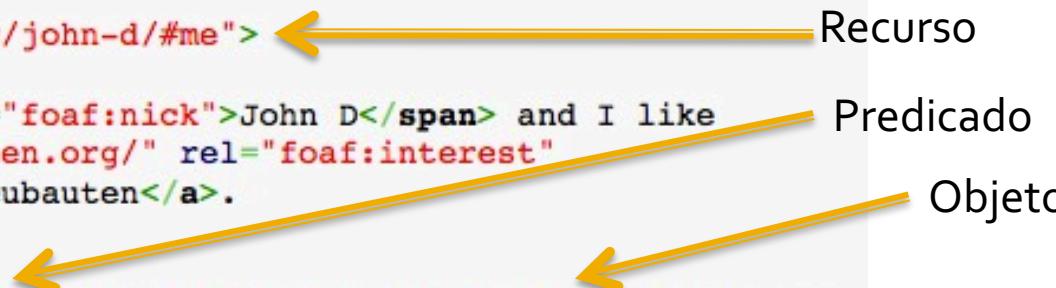
For example, in the line `favorite`, the `span` element is the Recurso, `property="foaf:interest"` is the Predicado, and `urn:ISBN:0752820907` is the Objeto.

RDF y RDFa

RDFa

■ HTML+RDFa

```
<html prefix="dc: http://purl.org/dc/elements/1.1/" lang="en">
  <head>
    <title>John's Home Page</title>
    <link rel="profile" href="http://www.w3.org/1999/xhtml/vocab" />
    <base href="http://example.org/john-d/" />
    <meta property="dc:creator" content="Jonathan Doe" />
    <link rel="foaf:primaryTopic" href="http://example.org/john-d/#me" />
  </head>
  <body about="http://example.org/john-d/#me">  Recurso
    <h1>John's Home Page</h1>
    <p>My name is <span property="foaf:nick">John D</span> and I like
      <a href="http://www.neubauten.org/" rel="foaf:interest"
         lang="de">Einstürzende Neubauten</a>.
    </p>
    <p>
      My <span rel="foaf:interest" resource="urn:ISBN:0752820907">favorite
      book is the inspiring <span about="urn:ISBN:0752820907"><cite
        property="dc:title">Weaving the Web</cite> by
        <span property="dc:creator">Tim Berners-Lee</span></span></span>.
    </p>
  </body>
</html>
```



RDF y RDFa

RDFa

■ Código equivalente RDF/XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:foaf="http://xmlns.com/foaf/0.1/"
           xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://example.org/john-d/">
    <dc:creator xml:lang="en">Jonathan Doe</dc:creator>
    <foaf:primaryTopic>
      <rdf:Description rdf:about="http://example.org/john-d/#me">
        <foaf:nick xml:lang="en">John D</foaf:nick>
        <foaf:interest rdf:resource="http://www.neubauten.org/" />
        <foaf:interest>
          <rdf:Description rdf:about="urn:ISBN:0752820907">
            <dc:creator xml:lang="en">Tim Berners-Lee</dc:creator>
            <dc:title xml:lang="en">Weaving the Web</dc:title>
          </rdf:Description>
        </foaf:interest>
      </rdf:Description>
    </foaf:primaryTopic>
  </rdf:Description>
</rdf:RDF>
```

Predicado → `rdf:about`

Recurso → `rdf:resource`

Objeto → `dc:title`

Schema.org

- Fundado por los grandes buscadores (Google, Microsoft, Yahoo y Yandex) para promocionar los datos estructurados en Internet
- Define una vocabulario para anotar los contenidos
- Es posible utilizar 3 sintaxis diferentes:
 - RDFa
 - Microdata
 - JSON-LD

Schema.org

RDFa

```
<div vocab="http://schema.org/" class="event-wrapper" typeof="Event">
  <div class="event-date" property="startDate" content="2013-09-
14T21:30">Sat Sep 14</div>
  <div class="event-title" property="name">Typhoon with Radiation
City</div>
  <div class="event-venue" property="location" typeof="Place">
    <span property="name">The Hi-Dive</span>
    <div class="address" property="address" typeof="PostalAddress">
      <span property="streetAddress">7 S. Broadway</span><br>
      <span property="addressLocality">Denver</span>,
      <span property="addressRegion">CO</span>
      <span property="postalCode">80209</span>
    </div>
  </div>
  <div class="event-time">9:30 PM</div>
  <span property="offers" typeof="Offer">
    <div class="event-price">
      <meta property="priceCurrency" content="USD" />$
      <meta property="price" content="13.00" />13.00
    </div>
    <a property="url"
      href="http://www.ticketfly.com/purchase/309433">Tickets</a>
    </span>
  </div>
```

Schema.org

Microdata

```
<div class="event-wrapper" itemscope itemtype="http://schema.org/Event">
  <div class="event-date" itemprop="startDate" content="2013-09-
14T21:30">Sat Sep 14</div>
  <div class="event-title" itemprop="name">Typhoon with Radiation
City</div>
  <div class="event-venue" itemprop="location" itemscope
itemtype="http://schema.org/Place">
    <span itemprop="name">The Hi-Dive</span>
    <div class="address" itemprop="address" itemscope
itemtype="http://schema.org/PostalAddress">
      <span itemprop="streetAddress">7 S. Broadway</span><br>
      <span itemprop="addressLocality">Denver</span>, <span
itemprop="addressRegion">CO</span> <span itemprop="postalCode">80209</span>
    </div>
  </div>
  <div class="event-time">9:30 PM</div>
  <div itemprop="offers" itemscope itemtype="http://schema.org/Offer">
    <div class="event-price" itemprop="price"
content="13.00">$13.00</div>
    <meta itemprop="priceCurrency" content="USD"><a itemprop="url"
href="http://www.ticketfly.com/purchase/309433">Tickets</a>
  </div>
</div>
```

Schema.org JSON-LD

JavaScript Object Notation for Linked Data

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Event",
  "location": {
    "@type": "Place",
    "address": {
      "@type": "PostalAddress",
      "addressLocality": "Denver",
      "addressRegion": "CO",
      "postalCode": "80209",
      "streetAddress": "7 S. Broadway"
    },
    "name": "The Hi-Dive"
  },
  "name": "Typhoon with Radiation City",
  "offers": {
    "@type": "Offer",
    "price": "13.00",
    "priceCurrency": "USD",
    "url": "http://www.ticketfly.com/purchase/309433"
  },
  "startDate": "2013-09-14T21:30"
}
</script>
```

SPARQL

HELLO
MY NAME IS

SPARQL

SPARQL

Definición

- SPARQL Protocol And RDF Query Language es un lenguaje de consulta pero también un protocolo (se implementa como servicio Web y viene descrito por un WSDL)
- Permite realizar consultas sobre información semántica (ternas RDF)
- Su sintaxis recuerda a la de SQL (no es casualidad)

SPARQL

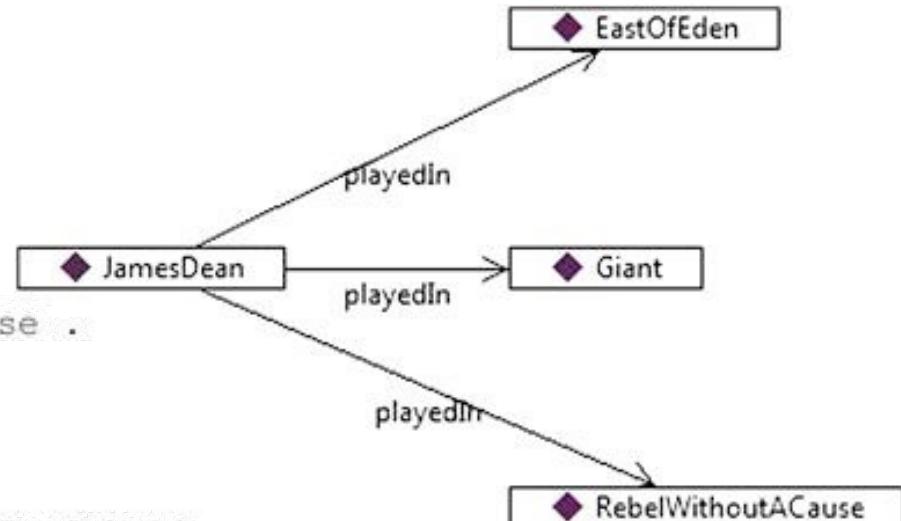
Patrones

Ejemplos de consultas

```
SELECT ?what WHERE { :JamesDean ?playedIn ?what . }
SELECT ?who WHERE { ?who :playedIn :Giant . }
SELECT ?what WHERE { :JamesDean ?what :Giant . }
```

Dichos patrones pueden ser vistos como patrones de grafos

Tell: :JamesDean :playedIn :Giant .
Tell: :JamesDean :playedIn :EastOfEden .
Tell: :JamesDean :playedIn :RebelWithoutACause .



Now if we ASK a question with SPARQL

Ask: SELECT ?what WHERE { :JamesDean :playedIn ?what }
Answer: :Giant, :EastOfEden, :RebelWithoutACause.

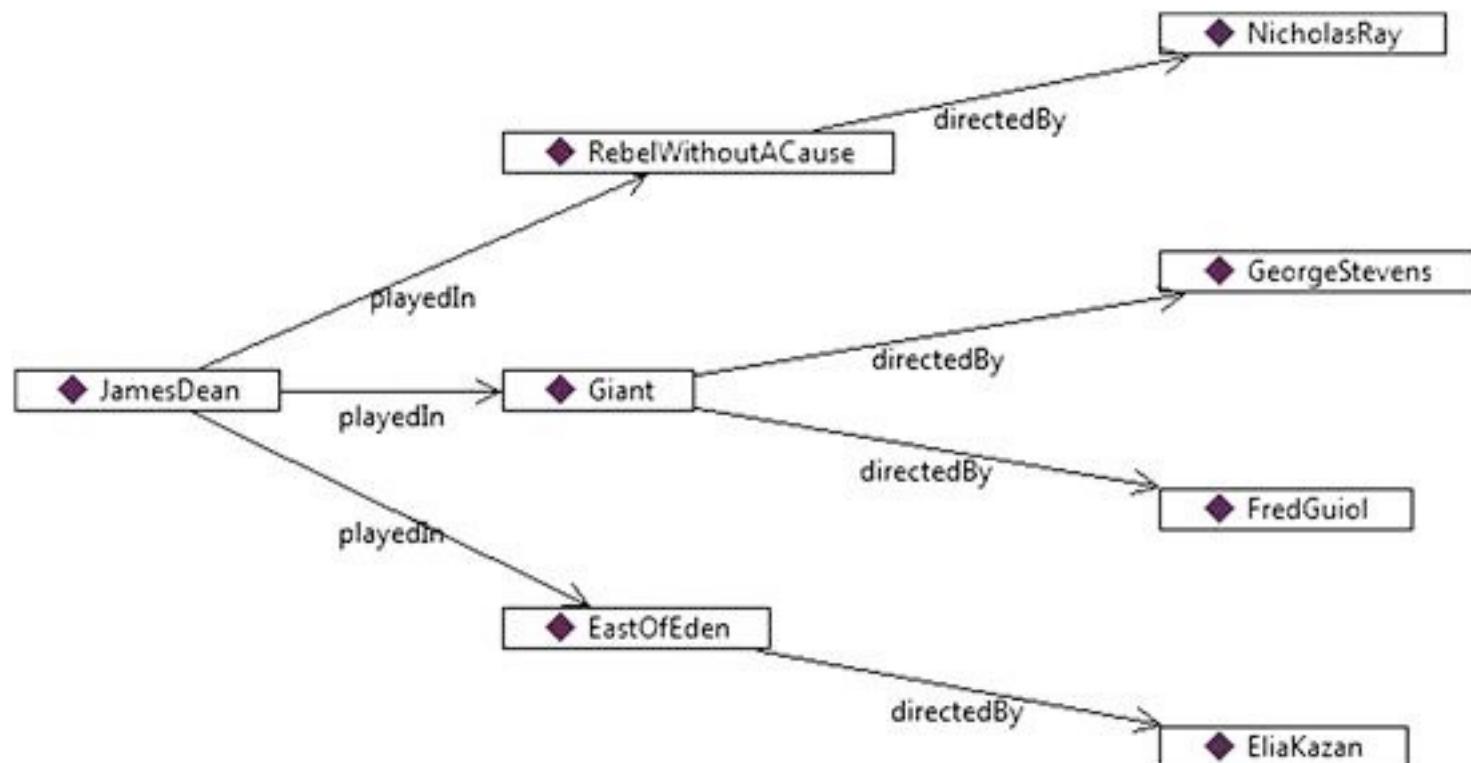
¿A qué lenguaje recuerda esto?



SPARQL Patrones

■ Ejemplos de consultas

```
:JamesDean :playedIn ?what .  
?what :directedBy ?who .
```



SPARQL

Selección

- Podemos preguntar por más de una variable

Ask:

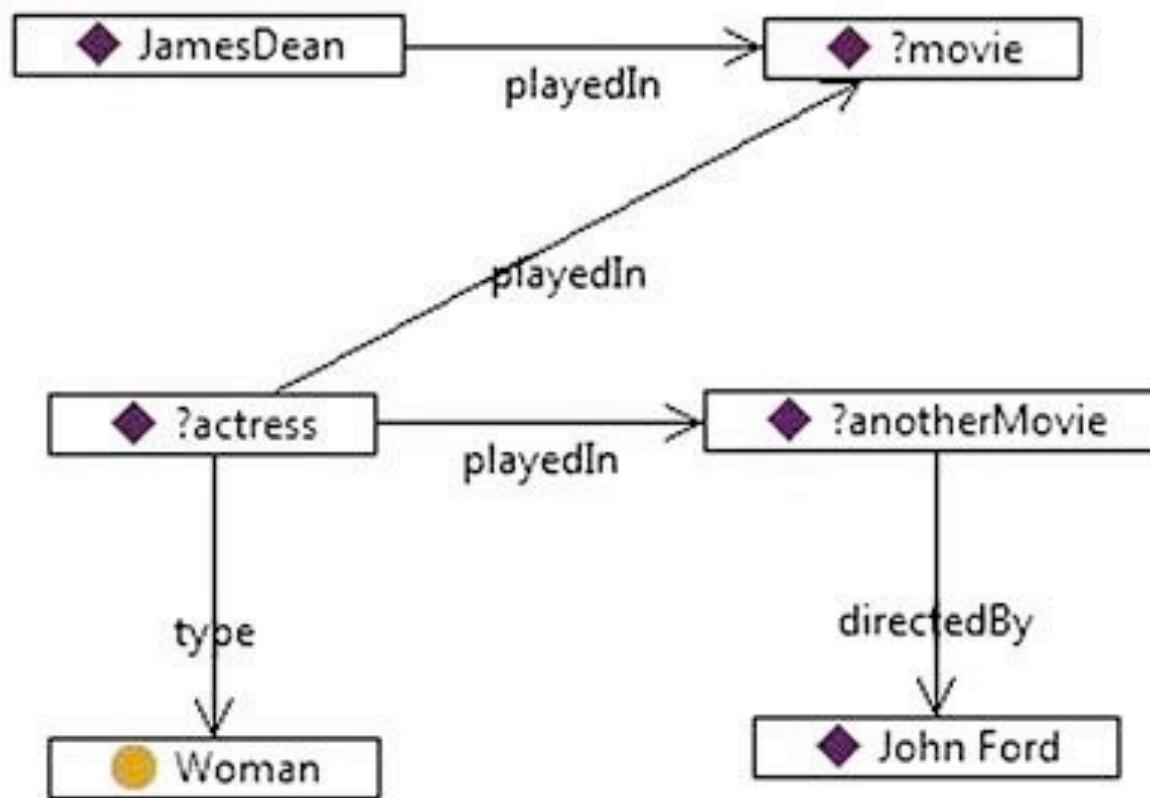
```
SELECT ?what ?who
WHERE { :JamesDean :playedIn ?what .
        ?what :directedBy ?who .}
```

?what	?who
:Giant	:GeorgeStevens
:Giant	:FredGuiol
:EastOfEden	:EliaKazan
:RebelWithoutaCause	:NicholasRay

SPARQL Selección

- Ejercicio: ¿cómo escribimos este patrón?

(a)



SPARQL

Orden

- El orden de los patrones importa

```
SELECT ?q3
WHERE (:JamesDean :playedIn ?q1 .
       ?q3 :playedIn ?q1 .
       ?q3 :playedIn ?q2 .
       ?q2 :directedBy :JohnFord .)
```

Más eficiente

```
SELECT ?q3
WHERE (?q3 :playedIn ?q1 .
       :JamesDean :playedIn ?q1 .
       ?q3 :playedIn ?q2 .
       ?q2 :directedBy :John Ford. )
```

Menos eficiente

¿No os sigue recordando ese lenguaje?

SPARQL

Consultando propiedades

- Se puede preguntar por cualquier elemento de las ternas

Ask:

```
SELECT ?property ?value  
WHERE { :JamesDean ?property ?value }
```

Answer:

?property	?value
bornOn	1931-02-08
diedOn	1955-09-30
playedIn	RebelWithoutaCause
playedIn	EastOfEden
playedIn	Giant
rdf:type	Man
rdfs:label	James Dean

SPARQL

Elementos distintos

- Es posible evitar repeticiones de elementos con DISTINCT

Ask:

```
SELECT ?property  
WHERE { :JamesDean ?property ?value}
```

Answer:

?property
bornOn
diedOn
playedIn
playedIn
playedIn
rdf:type
rdfs:label

Ask:

```
SELECT DISTINCT ?property  
WHERE { :JamesDean ?property ?value}
```

Answer:

?property
bornOn
diedOn
playedIn
rdf:type
rdfs:label

SPARQL

Filtros

- FILTER permite filtrar los resultados especificando una condición lógica

Ask:

```
SELECT ?actor
WHERE {?actor :playedIn :Giant .
        ?actor :diedOn ?deathdate .
        FILTER (?deathdate > "1961-11-24"^^xsd:date)}
```

Answer:

```
SELECT ?person
WHERE {?person a :Person .
        ?person :bornOn ?birthday .
        FILTER (?birthday > "Jan 1, 1960"^^xsd:date)
        FILTER (?birthday < "Dec 31, 1969"^^xsd:date)}
```

SPARQL

Patrones opcionales

- Los patrones opcionales no se tienen en cuenta para comprobar si casan las ternas pero sí para vincular variables

Ask:

```
SELECT ?actor ?deathdate
WHERE {?actor :playedIn :Giant .
      ?actor :diedOn ?deathdate .}
```

Answer:

actor	deathdate
RockHudson	1985-10-02
JamesDean	1955-10-30
...	

Ask:

```
SELECT ?actor ?deathdate
WHERE {?actor :playedIn :Giant .
      OPTIONAL {?actor :diedOn ?deathdate .}}
```

Answer:

Actor	deathdate
RockHudson	1985-10-02
JamesDean	1955-10-30
Elizabeth Taylor	(no binding)
...	

SPARQL

Construyendo conocimiento

- Podemos usar el resultado de una consulta para construir nuevas ternas

```
CONSTRUCT {?d rdf:type :Director .  
          ?d rdfs:label ?name . }  
WHERE {?any :directedBy ?d .  
       ?d rdfs:label ?name . }
```

- Esto permite aplicar reglas a la información semántica (inferencia, deducciones)

```
CONSTRUCT {?q1 :hasSon :q2 .}  
WHERE {?q2 :hasFather ?q1}
```

¿A qué nos recuerda esto?

SPARQL

Ejemplo

- Existen bases de datos de ternas RDF disponibles en la Web para ser consultadas
- También existen motores de consulta SPARQL
 - Linked Movie DataBase
 - <http://data.linkedmdb.org/>
 - Simple SPARQL client
 - <http://sparqlclient.eionet.europa.eu/>



Para ampliar conocimientos

- D. Allemang, J. Hendler, Semantic Web for the Working Ontologist, 2nd ed., Morgan Kaufmann, 2011
- Estándares de W3C para RDF: http://www.w3.org/standards/techs/rdf#w3c_all
- Estándares de W3C para OWL: http://www.w3.org/standards/techs/owl#w3c_all