Metadati: RDF e RDFS

Fabio Vitali

Introduzione

Esaminiamo:

- Meta informazioni e web semantico
- Il modello di RDF
- La sintassi di RDF
- Il Dublin Core
- RDF Schema
- Alcune riflessioni

Meta informazioni

Tutti gli ultimi sforzi del WWW hanno uno scopo: generare informazioni che non siano soltanto destinati alla lettura, ma che possano essere riutilizzati per applicazioni automatiche.

Non c'è niente in un documento HTML che indichi l'argomento trattato o la fonte delle informazioni. L'unico tipo di ricerca che si può fare su un documento è la ricerca sul contenuto. Questo non è sufficiente nella maggior parte delle volte: usando un motore di ricerca si ottiene un qualche migliaio di hit, la maggior parte dei quali non serve assolutamente a niente.

Le meta informazioni permettono agli autori di specificare informazioni sui loro documenti che siano non soltanto leggibili, ma anche interpretabili in maniera intelligente dalle applicazioni di rielaborazione, e soprattutto dai motori di ricerca.

L'utilizzo sistematico di meta-informazioni, ci dicono, ci porterà alla prossima generazione di Web: il Web semantico

Il Web semantico

Il W3C considera l'ideale evoluzione del Web dal machine-representable al machine-understandable. L'idea è di generare documenti che possano al tempo stesso essere letti ed apprezzati da esseri umani, ma anche acceduti ed interpretati da agenti automatici alla ricerca di contenuti.

Il Web si deve dunque dotare di una sovrastruttura semantica utilizzabile dalle applicazioni, in modo da poter svolgere quelle funzioni che oggi debbono essere fatte a mano o codificate dentro ai programmi.

Questo porta al *web semantico*, in cui non esprimo *testi* (all'interno dei quali le informazioni stanno nascoste e richiedono un umano), ma *affermazioni* (informazioni non ambigue, che esprimono relazioni tra oggetti, risorse, esseri umani, fatti del mondo reale, e che possono essere utilizzate anche da applicazioni automatiche).

Tre sono le tecnologie chiave per questo sviluppo:

- URI (+ XPointer), un meccanismo generico per identificare risorse
- XML, una meta-sintassi utilizzabile da ogni applicazione.
- RDF, un linguaggio per esprimere affermazioni

URI

Diventa qui importante specificare come gli URI non sono necessariamente URL (indirizzi accessibili di documenti Web), ma stringhe identificative di qualunque tipo di oggetto o concetto:

- http://www.cs.unibo.it/~fabio/index.html: un documento
- http://purl.org/dc/terms/modified: un concetto espresso in Dublin Core
- tag:sandro@world.std.org,2001-06-05:nome: un identificatore persistente associato ad un indirizzo e-mail ed una data
- esl:SHA1:iQAAwUBO51bkD6DK6KYhyiEEQJLqwCfSviAHvC1REXEkOGEWf9pAByCRwAni92xgCJqNL4fvrLRlGFK5szDXfckCE:nome: un identificatore *esl*, non modificabile, non falsificabile, non ripudiabile

N.B.: per qualunque identificatore vale una derivazione della legge di Zooko "buono, veloce, a buon mercato: scegline due", cioè la legge "sicuro, leggibile, decentralizzato: scegline due".

XML

Ottimo come sintassi e struttura dati:

- Elimina ambiguità tra contenuto e markup
- Elimina incertezze e dipendenze da specifiche codifiche carattere
- Fornisce API e modelli concettuali semplici per trattare qualunque tipo di struttura dati

Ma non perfettamente adatto per il Semantic Web:

- Troppi modi "linguistici" per esprimere gli stessi concetti
- Attributi e entità sono retaggio di un passato di linguaggio per documenti pensati per essere letti.

Meglio trovare un modello astratto per esprimere i concetti, e lasciare ad XML il compito di renderli in maniera linguistica.

RDF

Resource Description Framework è il modello astratto proposto dal W3C per esprimere affermazioni sul mondo.

RDF permette di esprimere ogni affermazione come una tripla (Soggetto, Predicato, Oggetto) (ad es.: "il documento http://www.host.org/~mrossi è stato creato da Mario Rossi"), dove il soggetto è un URI, il predicato esprime una relazione, e l'oggetto è un'altra risorsa, oppure un valore letterale.

Oltre alle affermazioni, RDF permette di esprimere anche citazioni, ovvero reificazioni, ovvero meta-affermazioni, vale a dire affermazioni su altre affermazioni (es.: "Andrea dice che il documento http://www.host.org/~mrossi è stato creato da Mario Rossi").

RDF NON è un formato XML, ma un modello astratto. Esistono però *linearizzazioni in XML* di RDF. Caratteristica di queste linearizzazioni è che NON sono univoche.

Esistono anche linearizzazioni non XML di RDF. Una di queste, Notation3, è stata introdotta da Berners-Lee, ed ha un certo successo.

RDF Schema

In RDF ogni predicato è astratto, senza connessioni né riferimenti, senza relazione con altri predicati.

RDF Schema permette di esprimere relazioni e vincoli tra predicati, permette di segnalare l'esistenza di proprietà caratteristiche di un concetto, che permettano di esprimere in maniera organizzata e sistematica affermazioni simili su risorse simili.

RDF Schema permette di specificare classi e proprietà, ed elencare le proprietà caratteristiche di una classe, e dominio e codominio di queste proprietà.

Ontologie, DAML+OIL e OWL

Il livello successivo è la possibilità di trarre conclusioni dalle affermazioni. RDF Schema è ancora troppo povero per permettere questo tipo di ragionamenti.

C'è bisogno di un linguaggio per esprimere inferenze, ovvero creazione di informazioni nuove attraverso la manipolazione automatica di informazioni già acquisite.

Una proposta di linguaggio per inferenze è DAML+OIL, che viene adesso convertito in OWL (Ontology Web Language) all'interno del W3C nel contesto RDF

DAML+OIL precisa l'esistenza di relazioni tra proprietà. Ad esempio, permette di dire che A e B sono l'una l'inverso dell'altra, o che C e D sono equivalenti.

Inoltre è necessario aggiungere un po' di ulteriori concetti (sostanzialmente i quantificatori universali ed esistenziali) per arrivare veramente ad un linguaggio dei predicati del primo ordine che permetta di esprimere vere e proprie inferenze sulle affermazioni RDF

Web of trust

Nel momento in cui inizio a realizzare inferenze, ho da considerare anche il problema della veridicità delle informazioni, e della loro affidabilità.

Che deduzioni posso ottenere dalla combinazione di due o più collezioni RDF, se tra di loro esistono affermazioni contraddittorie?

Una fragilità fondamentale dei sistemi di logica del primo ordine è data dal fatto che non solo le affermazioni contraddittorie non determinano nuova informazione, ma possono essere usate per generare qualunque inferenza: A □□A □ *.

Se una collezione RDF dice che il cielo è blu, e un'altra dice che il cielo non è blu, io sono logicamente autorizzato a concludere che 4 è dispari o che voi mi dovete €50 a testa.

Il passo successivo allora è creare una rete di affermazioni di affidabilità e fiducia (trust) sulle collezioni, in cui viene espresso il valore di affidabilità delle affermazioni contenute, e attraverso sistemi di sicurezza e crittografia rendere più affidabile il sistema di affermazioni .

Il web of trust è allora il passo ultimo per permettere la creazione di significato utile, automatico, affidabile su documenti e cose del mondo reale.

Il linguaggio RDF

E' una delle più importanti raccomandazioni del W3C.

Molti motori di ricerca stanno già usando RDF per descrivere il contenuto dei loro motori di ricerca. Tuttavia RDF ha senso se c'è attiva partecipazione da parte degli autori di siti, e ad oggi non esistono software ragionevoli per esprimere gradevolmente queste informazioni.

RDF è composto da due documenti:

- Model and Syntax Specification (W3C Recommendation del 25 marzo 2002): espone la struttura fondamentale del modello RDF, e descrive una possibile sintassi basata su XML.
- RDF Schema (W3C Candidate Recommendation 27 March 2000): espone la sintassi per definire schemi e vocabolari di metainformazioni.

Il modello di RDF

Il modello di RDF è basato su tre concetti:

- Risorse: tutto ciò che viene descritto. Ogni risorsa è indentificata da un URI; può quindi essere anche un oggetto non accessibile da web.
- Proprietà: un attributo che voglio associare alla risorsa. E' una coppia attributo-valore. Ogni proprietà ha un significato specifico, una serie di valori leciti, è associabile ad uno o più tipi di risorsa.
- Asserzioni (*statement*): l'associazione di una proprietà ad una risorsa. Ogni asserzione ha una struttura obbligata del tipo "soggetto", "predicato", "oggetto".

Soggetto (risorsa)	http://www.host.org/~mrossi
Predicato (proprietà)	Autore
Oggetto (letterale)	"Mario Rossi"

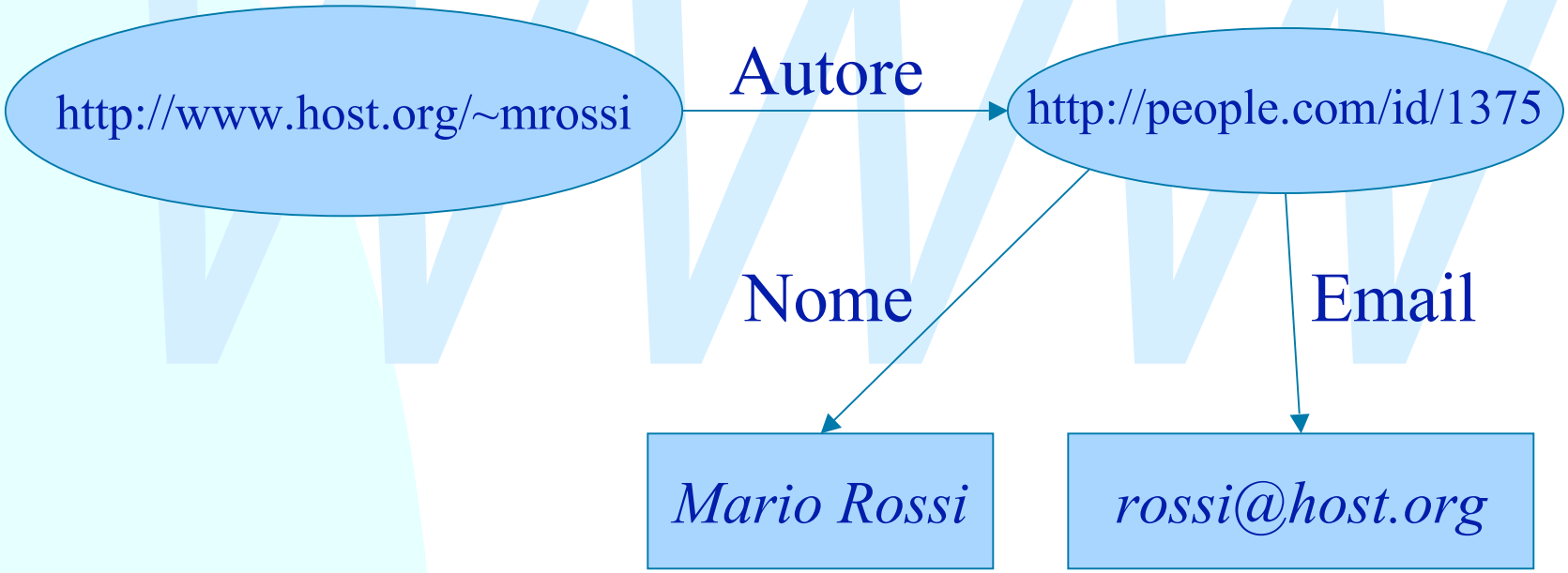
Rappresentazione grafica (1)

La proprietà "Autore" della risorsa "http://www.host.org/~mrossi" vale "Mario Rossi"

A seguire: Rappresentazione grafica (2)

Rappresentazione grafica (2)

La proprietà "Autore" della risorsa "http://www.host.org/~mrossi" è "Mario Rossi", che ha e-mail "rossi@host.org".



Sintassi estesa (1)

Il primo caso diventa in sintassi estesa:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi">
   <s:Autore>Mario Rossi</s:Autore>
</rdf:Description>
```

Il secondo caso:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi">
   <s:Autore rdf:resource=”http://people.com/id/1375"/>
</rdf:Description>

<rdf:Description rdf:about=”http://people.com/id/1375">
   <s:Nome>Mario Rossi</s:Nome>
   <s:Email>rossi@host.org</s:Email>
</rdf:Description>
```

Sintassi estesa (2)

Il secondo esempio è equivalente alla seguente forma:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi">
   <s:Autore>
      <rdf:Description rdf:about=”http://people.com/id/1375">
         <s:Nome>Mario Rossi</s:Nome>
         <s:Email>rossi@host.org</s:Email>
      </rdf:Description>
   </s:Autore>
</rdf:Description>
```

Tipizzazione

E’ possibile assegnare ad ogni risorsa un tipo appartenente ad uno schema di meta informazioni:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi">
  <s:Autore>
    <rdf:Description rdf:about=”http://people.com/id/1375">
      <rdf:type rdf:resource="/myschema.rdf#Persona"/>
      <s:Nome>Mario Rossi</s:Nome>
      <s:Email>rossi@host.org</s:Email>
    </rdf:Description>
  </s:Autore>
</rdf:Description>
```

L’attributo rdf:type specifica l’URI della definizione del tipo.

Prima sintassi abbreviata

Esistono alcune forme equivalenti ma più compatte.

Il primo esempio è equivalente alla seguente forma:

```
<rdf:Description
      rdf:about="http://www.host.org/~mrossi"
      s:Autore="Mario Rossi" />
```

I predicati (s:Autore) che hanno come oggetto elementi di tipo stringa (“Mario Rossi”) e non sono ripetuti vengono direttamente inseriti come attributi di “rdf:Description”.

A seguire: Seconda sintassi abbreviata

Seconda sintassi abbreviata

Applicando la seconda sintassi abbreviata al secondo esempio si ottiene:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi”>
   <s:Autore rdf:about=“http://people.com/id/1375”
      s:Nome=“Mario Rossi”
      s:Email=rossi@host.org/>
</rdf:Description>
```

E’ applicabile quando l’oggetto di un’asserzione è una risorsa le cui proprietà hanno come valore un letterale.

Terza sintasi abbreviata

E’ possibile usare il valore del tipo come predicato:

```
<rdf:Description about="http://www.host.org/~mrossi">
  <s:Autore>
    <s:Persona rdf:about=“http://people.com/id/1375”>
      <s:Nome>Mario Rossi</s:Nome>
      <s:Email>rossi@host.org</s:Email>
    </s:Persona>
  </s:Autore>
</rdf:Description>
```

Contenitori

A volte è importante fare riferimento ad un insieme di risorse (ad esempio, se un documento è stato creato da più autori, o se lo stesso autore ha fatto più di un documento, ecc.)

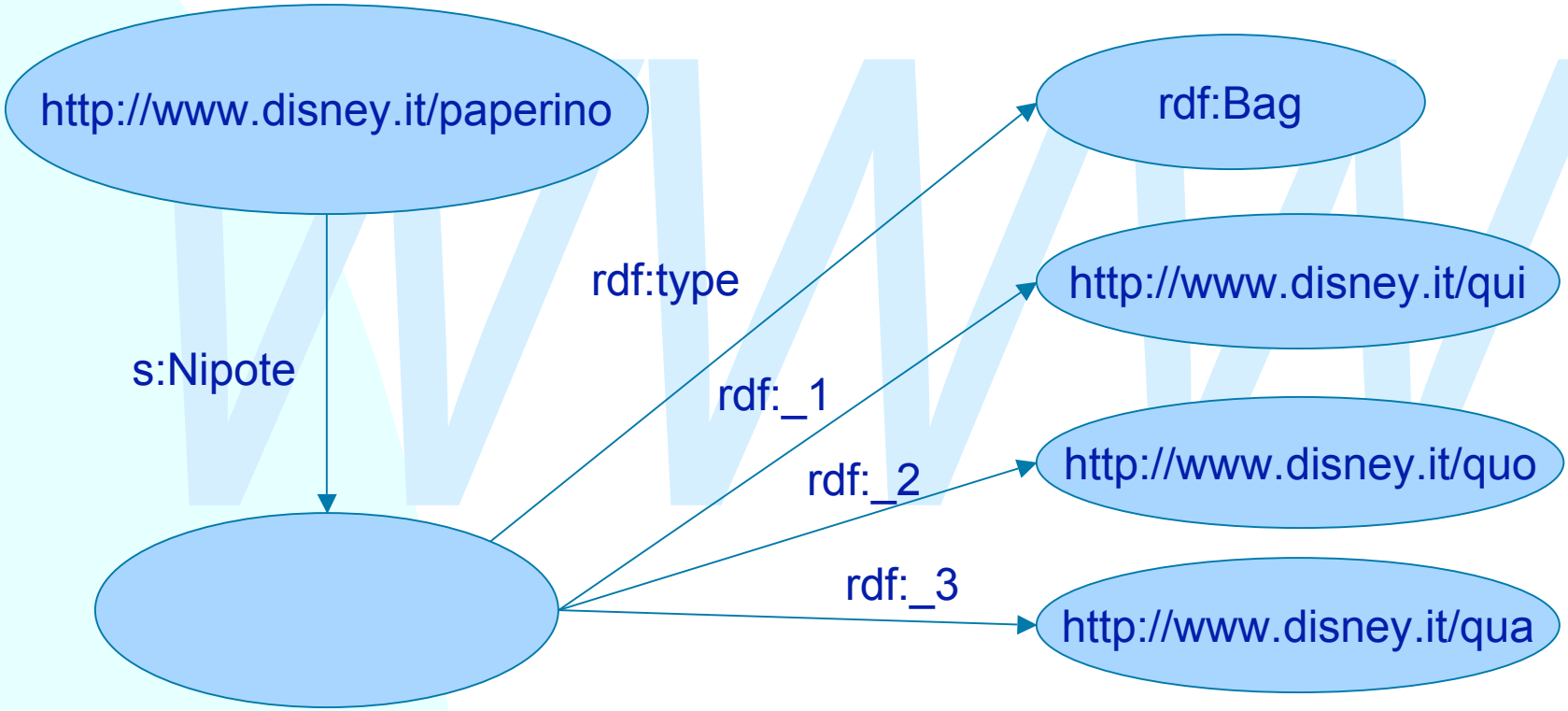
In questo caso tali risorse devono essere inserire all'interno di un contenitore che sarà l'oggetto dello statement.

RDF definisce tre tipi di contenitori:

- **Bag**. E' un insieme con ripetizioni. L'ordine non è rilevante.
- **Sequence**. E' un insieme con ripetizioni ed un ordine definito tra le risorse presenti.
- **Alternative**. E' un insieme senza ripetizioni in cui può essere scelto uno solo degli elementi. L'ordine degli elementi può essere usato per esprimere preferenza.

Rappresentazione dei contenitori

I nipoti di Paperino sono Qui, Quo, Qua.



Sintassi dei contenitori

```
<rdf:Description rdf:about="http://www.disney.it/paperino">
   <s:Nipote>
      <rdf:Description>
        <rdf:type rdf:resource=
          "http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag"/>
        <rdf:_1 rdf:resource="http://www.disney.it/qui"/>
        <rdf:_2 rdf:resource="http://www.disney.it/quo"/>
        <rdf:_3 rdf:resource="http://www.disney.it/qua"/>
      </rdf:Description>
   </s:Nipote>
</rdf:Description>
```

Analogamente si useranno i tipi rdf:Seq per le sequenze e rdf:Alt per le alternative.

Sintassi alternativa (1)

Se non si vuole sottolineare l'ordine degli elementi, è possibile usare l'elemento li. Inoltre, abbiamo raccolto la Description di tipo Bag in un elemento bag.

```
<rdf:Description rdf:about="http://www.disney.it/paperino">
   <s:Nipote>
      <rdf:Bag>
        <rdf:li rdf:about="http://www.disney.it/qui"/>
        <rdf:li rdf:about="http://www.disney.it/quo"/>
        <rdf:li rdf:about="http://www.disney.it/qua"/>
      </rdf:Bag>
   </s:Nipote>
</rdf:Description>
```

Reificazione (1)

Come è possibile fornire meta-informazioni su una meta-informazione? Ad esempio come posso esprimere la frase *«Andrea afferma che Mario Rossi è l'autore della risorsa "http://www.host.org/~mrossi"»*?

Questo in breve significa attribuire la proprietà *«afferma»* allo statement *«Mario Rossi è l'autore della risorsa "http://www.host.org/~mrossi"»*. Occorre pertanto considerare la meta-informazione come una risorsa da descrivere.

Questa procedura si chiama *reificazione* (riduzione ad oggetto) della asserzione (o statement). Dopo avere reificato l'asserzione potrò esprimere ulteriori proprietà su di essa.

A seguire: Reificazione (2)

Reificazione (2)

Sono equivalenti:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi">
    <s:Autore>Mario Rossi</s:Autore>
</rdf:Description>

<rdf:Description>
 <rdf:subject rdf:resource="http://www.host.org/~mrossi"/>
 <rdf:predicate rdf:resource="/myschema.rdf#Autore"/>
 <rdf:object>Mario Rossi</rdf:object>
 <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#Statement"/>
</rdf:Description>
```

Reificazione (3)

Uno statement reificato può essere usato come oggetto di un altro predicato:

```
<rdf:Description>
 <rdf:subject rdf:resource="http://www.host.org/~mrossi"/>
 <rdf:predicate rdf:resource="/myschema.rdf#Autore"/>
 <rdf:object>Mario Rossi</rdf:object>
 <rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax-ns#Statement"/>
 <s:AffermatoDa>Andrea</s:AffermatoDa>
</rdf:Description>
```

Sintassi compatta di reificazione

L'attributo rdf:bagID permette di considerare uno statement esplicito come se fosse reificato.

La stessa description può avere un rdf:ID, utile per indicarla come fonte di meta-informazioni, ed un rdf:bagID utile per esprimere meta-informazioni su di essa.

La frase precedente è equivalente a:

```
<rdf:Description rdf:about="http://www.host.org/~mrossi"
                 rdf:bagID="R_001">
   <s:Autore>Mario Rossi</s:Autore>
</rdf:Description>

<rdf:Description rdf:about="#R_001">
   <s:AffermatoDa>Andrea</s:AffermatoDa>
</rdf:Description>
```

Dublin Core (1)

Il Dublin Core è uno schema di meta informazioni ideato per assegnare etichette ragionevoli alle risorse della rete.

Si chiama Dublin Core perché è considerato il nucleo (core) delle meta-informazioni interessanti per qualunque risorsa, e perché è nato da un'iniziativa di bibliotecari, archivisti, fornitori di contenuto e esperti di markup svoltasi nel 1995 a Dublino.

Dublin Core è indipendente da qualunque sintassi, ma ben si adatta a RDF.

Dublin Core versione 1 ha introdotto esattamente quindici categorie di meta-informazioni utili per la catalogazione di risorse di rete.

La versione 2 ha aggiunto un meccanismo di sottoclassi (detti *qualificatori*) delle categorie, ed ha introdotto un elenco iniziale di qualificatori.

A seguire: Dublin Core (2)

Dublin Core (2)

Le quindici categorie descrivono meta-informazioni di tre tipi:

Contenuto	**Proprietà intellettuale**	**Istanza**
Title	Creator	Date
Subject	Publisher	Format
Description	Contributor	Identifier
Type	Rights	Language
Source		
Relation		
Coverage		

I qualificatori permettono di specificare ulteriormente informazioni di queste categorie, secondo questi criteri:

- **Raffinamento dello schema**: fornisce alcuni significati più precisi sui termini. Ad esempio, “Date” ha come qualificatori: “created”, “valid”, “available”, “issued”, “modified”).
- **Supporto per codifiche specifiche**: permette di usare i valori di particolari codifiche all’interno del Dublin Core. Ad esempio, “Subject” ha come qualificatori: “LCSH” (Library of Congress Subject Headings), “MeSH” (Medical Subject Headings), “DDC” (Dewey Decimal Classification), ecc.

Un esempio di Dublin Core in RDF

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:dc="http://purl.org/metadata/dublin_core#">
  <rdf:Description rdf:about="http://www.dlib.org">
    <dc:Title>D-Lib Program</dc:Title>
    <dc:Description>
      The D-Lib program supports the community of people
      with research interests in digital libraries and
      electronic publishing.
    </dc:Description>
    <dc:Publisher>
      Corporation For National Research Initiatives
    </dc:Publisher>
    <dc:Date>1995-01-07</dc:Date>
    <dc:Subject><rdf:Bag>
      <rdf:_1>Research; statistical methods</rdf:_1>
      <rdf:_2>Education, research, related topics</rdf:_2>
      <rdf:_3>Library use Studies</rdf:_3>
    </rdf:Bag></dc:Subject>
    <dc:Type>World Wide Web Home Page</dc:Type>
    <dc:Format>text/html</dc:Format>
    <dc:Language>en</dc:Language>
  </rdf:Description>
</rdf:RDF>
```

RDF Schema 1.0

Il modello di RDF non permette di effettuare ***validazione*** di un valore o ***restrizione*** di un dominio di applicazione di una proprietà. Questo compito è svolto da *RDF Schema*.

A differenza di XML Schema o di un DTD, RDF Schema non vincola la struttura del documento, ma fornisce informazioni utili all'interpretazione del documento stesso.

RDF Schema fornisce un meccanismo di base per un sistema di tipizzazione da utilizzare in modelli RDF.

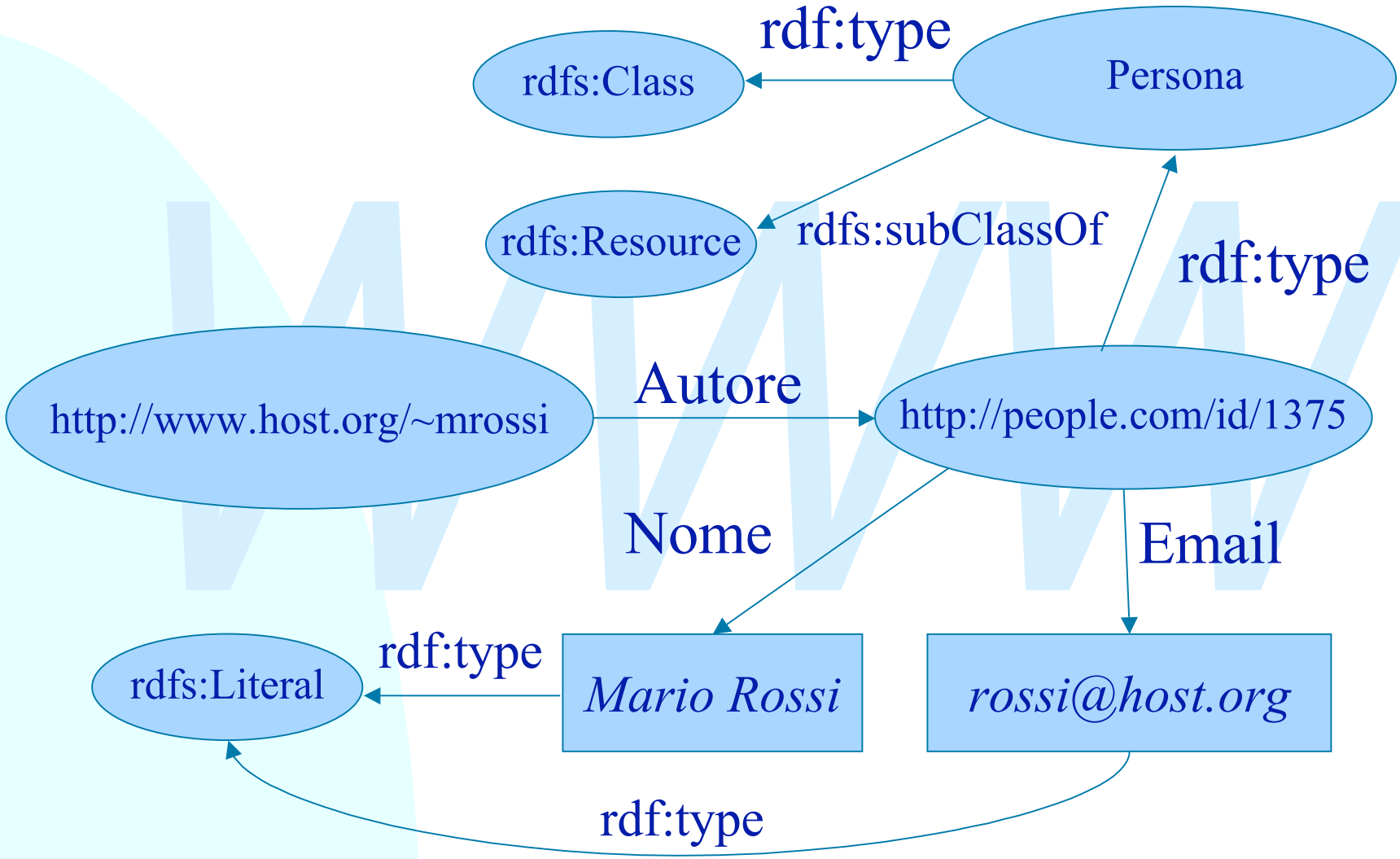
Lo schema è definito in termini di RDF stesso.

RDF Schema definisce un insieme di risorse RDF da usare per descrivere caratteristiche di altre risorse e proprietà RDF.

Le classi e le proprietà (1)

- *rdfs:Resource* Tutto ciò che viene descritto in RDF è detto *risorsa*. Ogni risorsa è istanza della classe rdfs:Resource.
- *rdfs:Literal* Sottoclasse di rdfs:Resource, rappresenta un letterale, una stringa di testo.
- *rdf:Property* Rappresenta le proprietà. E' sottoclasse di rdfs:Resource.
- *rdfs:Class* Corrisponde al concetto di *tipo* e di *classe* della programmazione object-oriented. Quando viene definita una nuova classe, la risorsa che la rappresenta deve avere la proprietà rdf:type impostata a rdfs:Class.
- *rdfs:subClassOf* Specifica la relazione di ereditarietà fra classi. Questa proprieta può essere assegnata solo a istanze di rdfs:Class. Una classe può essere sottoclasse di una o più classi (*ereditarietà multipla*).

Le classi e le proprietà (2)



Le classi e le proprietà (3)

```
<rdf:Description rdf:ID="Autoveicolo">
 <rdf:type
       rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf
       rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
</rdf:Description>

<rdf:Description rdf:ID="VeicoloPasseggeri">
 <rdf:type
       rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Autoveicolo"/>
</rdf:Description>
```

Le classi e le proprietà (4)

```
<rdf:Description rdf:ID="Van">
 <rdf:type
          rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Autoveicolo"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
 <rdf:type
          rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Van"/>
 <rdfs:subClassOf rdf:resource="#VeicoloPasseggeri"/>
</rdf:Description>
```

Le classi e le proprietà (4)

- *rdfs:subPropertyOf* Istanza di rdf:Property, è usata per specificare che una proprietà è una specializzazione di un'altra. Ogni proprietà può essere la specializzazione di zero o più proprietà.
- *rdfs:seeAlso* Specifica una risorsa che fornisce ulteriori informazioni sul soggetto dell'asserzione.
- *rdfs:isDefinedBy* E' sottoproprietà di rdfs:seeAlso e indica una risorsa che definisce il soggetto di un'asserzione

Le classi e le proprietà (5)

```
<rdf:Description rdf:ID="Genitore">
 <rdf:type
   rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
</rdf:Description>

<rdf:Description rdf:ID="Padre">
 <rdf:type
   rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
 <rdfs:subPropertyOf rdf:resource="#Genitore"/>
</rdf:Description>
```

I vincoli (1)

I predicati più utilizzati per esprimere vincoli su altre proprietà sono i seguenti:

- rdfs:range (codominio) Usato come predicato di una risorsa r, indica le classi che saranno oggetto di un'asserzione che ha r come predicato.
- rdfs:domain (dominio) Usato come predicato di una risorsa r, indica le classi (soggetto) a cui può essere applicata r.

A seguire: I vincoli (2)

I vincoli (2)

```
<rdf:Description rdf:ID="RegistratoA">
 <rdf:type
  rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
 <rdfs:domain rdf:resource="#Autoveicolo"/>
 <rdfs:range rdf:resource="#Persona"/>
</rdf:Description>

<rdf:Description rdf:ID="NumeroPasseggeri">
 <rdf:type
  rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#Property"/>
 <rdfs:domain rdf:resource="#VeicoloPasseggeri"/>
 <rdfs:range
rdf:resource="http://www.w3.org/2000/03/example/classes#
Number"/>
</rdf:Description>
```

Riflessioni (1)

Il Semantic Web deve integrare e sostituire il web normale per esprimere concetti ed informazioni in maniera comprensibile alle applicazioni.

Ci sono quattro stadi previsti:

- Espressione di affermazioni (RDF)
- Espressione di schemi sulle proprietà affermabili (RDF Schema)
- Espressione di relazioni tra le proprietà in ontologie (OWL)
- Espressione di livelli di affidabilità tra ontologie (Web of trust)

Ci sono varie riflessioni al proposito:

- Il principio della minima potenza
- Data scaping vs. form
- Il Pedantic Web
- Completezza e compatibilità

A seguire: Riflessioni (2)

Riflessioni (2)

Il principio della minima potenza

- Ad ogni livello vengono forniti meccanismi minimali per esprimere affermazioni al livello corrispondente: affermazioni e reificazioni al I, tipi e classi al II, opposti e equivalenze al III, selezione e coerenza al IV. Mettere tutto insieme spaventerebbe subito tutti per la complessità richiesta

Complessità del modello

- E comunque la gente è spaventata lo stesso: troppe cose da capire, troppe incertezze su formati, esprimibilità di concetti, ecc.

Data scaping vs. form

- Come generare automaticamente le affermazioni? Due approcci:
  - Data scraping: generazione automatica di affermazioni sulla base di regolarità dei dati
  - Form: applicazioni interattive che interrogano un esperto di dominio e rendono in maniera formale le sue conoscenze

Riflessioni (3)

Il Pedantic Web

- Non è forse un po' troppo? Confusioni naturali esistono (una persona con il suo nome, un documento con il suo titolo)
-
```
<rdf:Description
rdf:about="http://www.host.org/~mrossi">
  <s:Autore>Mario Rossi</s:Autore>
</rdf:Description>
```
- Come diavolo fa una stringa ad essere autore di un documento? Se nella mia applicazione mi bastano stringhe, perché, in nome di quale vantaggio economico, debbo introdurre nuovi concetti? E quando mi debbo interrompere?
- Non sa tutto molto di accademia, e di tentativi di classificazione del mondo? Non abbiamo già visto quanto presuntuoso, irrangiungibile e in ultima analisi futile siano questi tentativi?

Riflessioni (4)

Completezza e compatibilità

- Quello che bloccò molti progetti di intelligenza artificiale nei primi anni 90 fu il problema della compatibilità tra ontologie sviluppate indipendentemente.
- Non è solo un problema di trust, ma anche di pura e semplice confrontabilità dei modelli. Sono possibili innumerevoli organizzazioni dei dati, tutte lecite, tutte incompatibili.
- E inoltre ogni volta ci si accorge di aver lasciato fuori qualcosa, o di dover meglio specificare alcuni aspetti. Debuggare completamente un programma è difficile, un'ontologia impossibile.

In definitiva:

- Siamo sicuri che ne valga la pena?

Software utili

- Validatore e visualizzatore di documenti RDF: http://www.w3.org/RDF/Validator/
- Editor di documenti per il Dublin Core: http://www.ukoln.ac.uk/metadata/dcdot/
- Parser Java: http://www.hpl.hp.com/semweb/
- Parser Perl: http://www.w3.org/1999/02/26-modules/

Riferimenti

- Tim Berners-Lee. *Semantic Web* - XML2000 http://www.w3.org/2000/Talks/1206-xml2k-tbl
- Tim Berners-Lee, James Hendler, and Ora Lassila. *The Semantic Web,* http://www.scientificamerican.com/2001/0501issue/0501-berners-lee.html.
- Ora Lassila, Ralph R. Swick, *Resource Description Framework (RDF), Model and Syntax Specification*, W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax
- Dave Beckett, *RDF/XML Syntax Specification (Revised)*, W3C Recommendation 25 March 2002, http://www.w3.org/TR/2002/WD-rdf-syntax-grammar-20020325
- Dan Brickley, R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0*, W3C Candidate Recommendation 27 March 2000, http://www.w3.org/TR/rdf-schema
- Patrick Hayes, *RDF Model Theory*, http://www.w3.org/TR/2002/WD-rdf-mt-20020214

Fine Presentazione