**Il Web Semantico: Un’introduzione**

Questo documento è concepito come una pubblicazione introduttivo semplice ma completo per chiunque cercando di entrare nel Web Semantico: dai principianti fino ai lunghi hacker tempo. Pre-lettura consigliata: [il Semantic Web in Larghezza](http://logicerror.com/semanticWeb-long) .

**Sommario**

1. [Qual è il Web Semantico?](http://infomesh.net/2001/swintro/#whatIsSw)
2. [Semplice Modellazione dati: Schemata](http://infomesh.net/2001/swintro/#simpleData)

**Qual è il Web Semantico?**

Il Web semantico è una rete di informazioni collegate in modo tale da essere facilmente lavorabile da macchine, su scala globale. Si può pensare ad esso come un modo efficace di rappresentare dati sul World Wide Web, o come un database collegato a livello globale.

Il Semantic Web è stato ideato da Tim Berners-Lee, inventore del WWW, URI, HTTP e HTML. C'è un team dedicato di persone al World Wide Web Consortium ( [W3C](http://www.w3.org/) ) impegnate a migliorare, ampliare e uniformare il sistema, e molte lingue, pubblicazioni, strumenti e così via sono già state sviluppate. Tuttavia, le tecnologie del Semantic Web sono ancora molto in loro infanzie, e anche se il futuro del progetto in generale sembra essere brillante, sembra che ci sia poco consenso sulla probabile direzione e le caratteristiche dei primi Semantic Web.

Qual è il razionale per un tale sistema? I dati che si geneally nascosto in file HTML è spesso utile in alcuni contesti, ma non in altri. Il problema con la maggior parte dei dati sul Web che è in questa forma al momento è che è difficile da usare su larga scala, in quanto non esiste un sistema globale per la pubblicazione dei dati in modo tale che possa essere facilmente trattati da chiunque. Ad esempio, basta pensare di informazioni su eventi sportivi locali, informazioni meteo, orari aerei, statistiche Major League di baseball, e guide televisive ... tutte queste informazioni è presentato da numerosi siti, ma tutto in HTML. Il problema di questo è che, è alcuni contesti, è difficile utilizzare questi dati nei modi che si potrebbe desiderare di farlo.

Così il Web semantico può essere visto come un enorme soluzione ingegneristica ... ma è più. Ci troveremo che in quanto diventa più facile per pubblicare i dati in una forma repurposable, così che più persone vorranno pubish dati, e ci sarà un knock-on o l'effetto domino. Possiamo trovare che un gran numero di applicazioni Web Semantico può essere utilizzato per una varietà di compiti diversi, aumentando la modularità delle applicazioni sul Web. Ma basta ragionamento soggettivo ... su come questo sarà compiuto.

Il Semantic Web è generalmente costruita su sintassi che utilizzano URI per rappresentare i dati, di solito in triple strutture fondate: Ie molti triple di dati URI che possono essere detenute in banche dati, o scambiati sul world wide web utilizzando una serie di particolari sintassi sviluppati appositamente per il compito. Questi sono chiamati sintassi sintassi "Resource Description Framework".

**URI - Uniform Resource Identifier**

Un URI è semplicemente un identificatore Web: come le stringhe che iniziano con "http:" o "ftp:" che spesso si trovano sul World Wide Web. Chiunque può creare un URI, e la proprietà di loro è chiaramente delegata, in modo da formare una tecnologia base ideale con cui costruire un Web globale in cima. Infatti, il World Wide Web è una cosa: tutto ciò che ha un URI è considerato "sul Web".

La sintassi di URI è attentamente regolata dalla IETF, che ha pubblicato [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt) come specifica generale URI. Il W3C mantiene un [elenco di schemi URI](http://www.w3.org/Addressing/schemes) .

**RDF - Resource Description Framework**

Un triplo può semplicemente essere descritto come tre URI. Un linguaggio che utilizza tre URI in modo viene chiamato RDF: W3C hanno sviluppato una serializzazione XML di RDF, la "sintassi" nel [modello RDF ​​e la raccomandazione sintassi](http://www.w3.org/TR/1999/REC-rdf-syntax-19990222) . XML RDF è considerato essere il formato standard per interscambio RDF ​​sul Web semantico, anche se non è l'unico formato. Ad esempio, Notation3 (che saremo attraversando più avanti in questo articolo) è un ottimo testo semplice serializzazione alternativo.

Una volta informazioni è in forma RDF, diventa facile da lavorare, dato RDF è un formato generico, che ha già molti parser. XML RDF è una specifica piuttosto prolisso, e può richiedere un po 'per abituarsi (ad esempio, per imparare XML RDF correttamente, è necessario capire un po' di XML e gli spazi dei nomi in anticipo ...), ma diamo un rapido sguardo a un esempio di XML RDF momento: -

<Rdf: RDF xmlns: rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"

xmlns: dc = "http://purl.org/dc/elements/1.1/"

xmlns: foaf = "http://xmlns.com/0.1/foaf/">

<Rdf: Description rdf: about = "">

<Dc: creator rdf: parseType = "Resource">

<Foaf: name> Sean B. Palmer </ foaf: name>

</ Dc: creator>

<Dc: title> Il Web Semantico: An Introduction </ dc: title>

</ Rdf: Description>

</ Rdf: RDF>

Questo pezzo di RDF dice in sostanza che l'articolo ha il titolo "Il Web Semantico: An Introduction", ed è stato scritto da qualcuno il cui nome è "Sean B. Palmer". Qui ci sono le triple che questo RDF produce: -

<> <Http://purl.org/dc/elements/1.1/creator> \_: x0.

questo <http://purl.org/dc/elements/1.1/title> "The Semantic Web: An Introduction".

\_: X0 <http://xmlns.com/0.1/foaf/name> "Sean B. Palmer".

Questo formato è in realtà un testo serializzazione pianura di RDF chiamato "Notation3", che saremo coprendo [in seguito](http://infomesh.net/2001/swintro/#notation3) . Si noti che alcune persone in realtà preferiscono utilizzare RDF XML per Notation3, ma è generalmente accettato che Notation3 è più facile da usare, ed è ovviamente trasformabile in XML RDF comunque.

**Perché RDF?**

Quando le persone si confrontano con RDF XML per la prima volta, di solito hanno due domande: "perché usare RDF piuttosto che XML?", E "usiamo schema XML in combinazione con RDF?".

La risposta alla domanda "perché usare RDF piuttosto che XML?" è molto semplice, ed è duplice. In primo luogo, il beneficio che si ottiene dalla stesura di una lingua in RDF è che le informazioni mappe *direttamente* e *senza ambiguità* ad un modello, un modello che è decentrata, e per la quale ci sono molti parser generico già disponibili.Ciò significa che quando si dispone di un'applicazione RDF, si sa che i bit di dati sono la semantica della domanda, e che i bit sono fluff solo sintattico. E non solo si sa che,*tutti* sanno che, spesso implicitamente senza nemmeno leggere una specifica perché RDF è così ben conosciuto. La seconda parte della risposta duplice è che speriamo che i dati RDF diventeranno una parte del Semantic Web, in modo che i benefici di redigere i tuoi dati in RDF, attira ora paralleli di redigere le informazioni in HTML nei primi giorni del Web.

La risposta a "fare usiamo schema XML in combinazione con RDF?" è quasi il più breve. XML Schema è un linguaggio per limitare la *sintassi* di applicazioni XML. RDF ha già costruito in BNF che precisa come il linguaggio deve essere utilizzato, così sulla faccia di esso la risposta è un solido "no". Tuttavia, utilizzando XML Schema unitamente RDF *può* essere utile per la creazione di tipi di dati e così via. Pertanto la risposta è "possibilmente", con un avvertimento che non è usata per controllare la sintassi di RDF. Si tratta di un malinteso comune, perpetuato per troppo tempo ormai.

**Screen scraping, e Forme**

Per il Web Semantico di raggiungere il suo pieno potenziale, molte persone hanno bisogno di avviare la diffusione di dati RDF. Dove si trova questa informazione sta per venire? Molti può derivare da molte pubblicazioni dati che esistono oggi, utilizzando un processo chiamato "raschiatura screen". Scraping schermo è l'atto di letteralmente ottenere i dati da una fonte in una forma più gestibile (cioè RDF) con qualsiasi mezzo sono a portata di mano. Due strumenti utili per screen scraping sono XSLT (un linguaggio trasformazioni XML), e espressioni regolari (in Perl, Python, e così via).

Tuttavia, screen scraping è spesso una soluzione noioso, quindi un altro modo per accedere è di costruire sistemi adeguati RDF che prendono input da parte dell'utente e quindi memorizzarlo subito in RDF. Tali dati, è possibile inserire al momento della registrazione per un nuovo account di posta, l'acquisto di alcuni CD on-line, o alla ricerca di una macchina usata possono tutti essere memorizzati come RDF e poi utilizzati sul Web Semantico.

**Notation3: RDF Made Easy**

Come avrete visto sopra, XML RDF può essere piuttosto difficile, ma per fortuna, ci si sono forme di insegnamento più semplici di RDF. Uno di questi si chiama "Notation3", ed è stato sviluppato da Tim Berners-Lee. C'è un po 'di documentazione che copre N3, tra cui una [specifica](http://www.w3.org/DesignIssues/Notation3) , e un ottimo [Primer](http://www.w3.org/2000/10/swap/Primer) .

I criteri di progettazione dietro Notation3 erano abbastanza semplice: progettare un semplice, facile da imparare formato scribblable RDF, che è facile da analizzare e costruire applicazioni di grandi dimensioni in cima. In Notation3, possiamo semplicemente scrivere gli URI in una tripla, li delimitando con un "<" e ">" simboli. Ad esempio, ecco una semplice triplo composto da tre URI triplica: -

<Http://xyz.org/#a> <http://xyz.org/#b> <http://xyz.org/#c>.

Per utilizzare valori letterali, semplicemente racchiudere il valore tra doppie virgolette, quindi: -

<Http://xyz.org/#Sean> <http://xyz.org/#name> "Sean".

Se non si vuole dare un URI per qualcosa che si sta parlando, allora c'è un concetto anche per questo (questo è come dire "c'è qualcuno chiama ... ma senza dare loro un URI). È possibile utilizzare semplicemente un trattino e due punti, e poi mettere un po 'di etichetta lì: -

\_: A1 <http://xyz.org/#name> "Sean".

Questo può essere letto come "c'è qualcosa che ha il nome di Sean", o "a1 ha il nome Sean, per un valore di a1". Queste cose sono chiamati nodi anonimi, perché non hanno un URI, e sono a volte indicato come nodi esistenzialmente quantificati.

Nota come in uno degli esempi di cui sopra, abbiamo utilizzato l'URI "http://xyz.org/#" tre volte, con solo l'ultimo carattere cambia ogni volta? Notation3 ci dà un ottimo modo per abbreviare questo: dando parti di URI alias, e l'utilizzo di tali alias invece. Ecco come si dichiara un alias in Notation3: -

prefix xyz: <http://xyz.org/#>.

Si noti che si deve sempre dichiarare un alias prima di poterlo utilizzare. Per utilizzare un alias, è sufficiente utilizzare la "xyz:" bit al posto del URI, e *non* avvolgere il termine causando la "<" e ">" delimitatori. Ad esempio, invece di scrivere: -

<Http://xyz.org/#a> <http://xyz.org/#b> <http://xyz.org/#c>.

Possiamo invece fare: -

prefix xyz: <http://xyz.org/#>.

: A: b: c.

Si noti che non importa quello che alias si usa per un URI, fino a quando si utilizza lo stesso in tutto il documento. È inoltre possibile dichiarare molti alias. I seguenti bit di codice sono sia equivalente al pezzo di codice sopra: -

prefix blargh: <http://xyz.org/#>.

blargh: un blargh: b blargh: c.

prefix blargh: <http://xyz.org/#>.

prefix xyz: <http://xyz.org/#>.

blargh: un xyz: b blargh: c.

Tuttavia, va notato che spesso usiamo qualche alias praticamente come standard, in modo che quando gli sviluppatori del Web Semantico scambiano codice in formato testo, possono lasciare i prefissi fuori e la gente può indovinare che cosa stanno parlando. Si noti che il codice dovrebbe *non* implementare questa funzione. Ecco un esempio di alcuni alias "standard": -

prefix: <#>.

prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

prefix DAML: <http://www.daml.org/2001/03/daml+oil#>.

logprefix: <http://www.w3.org/2000/10/swap/log#>.

prefix dc: <http://purl.org/dc/elements/1.1/>.

prefix foaf: <http://xmlns.com/foaf/0.1/>.

L'alias vuota ":" è spesso usato per indicare un nuovo spazio dei nomi che l'autore non ha ancora creato un URI per (tut, tut). Lo usiamo in questa introduzione.

Notation3 ha molti altri piccoli costrutti compresi contesti, elenchi DAML, e modi alternativi di rappresentare nodi anonimi, ma non abbiamo bisogno di preoccuparci di loro qui. Si noti che una sintassi è stato concepito per essere un sottoinsieme ancora più semplice di Notation3, chiamato [N-triple](http://www.w3.org/2001/sw/RDFCore/ntriples/) , ma non usa prefissi, e quindi molti degli esempi in questo articolo non sono validi N-triple, ma sono valide Notation3.

Dan Connolly volta chiamato Notation3 "RDF strumento di authoring di poveri-man" un (fonte: tronchi RDF IG, 2001-06-01 03:55:12). A quanto pare, si chiama Notation3 perché "RDF M & S è stato il primo, l'uomo di paglia RDF è stato il secondo, e questo è il terzo" (fonte: [RDF IG F2F 2001/02/26](http://logicerror.com/w3c-meeting-2001-2-26) ).

**CWM: Un RDF XML E Notation3 Inference Engine**

Anche se non discuteremo motori di inferenza fino più avanti in questo articolo, dobbiamo notare a questo punto che molto RDF ​​e la lavorazione del Semantic Web (anche se spesso solo sperimentale o dimostrativo, in questa fase) è fatto con un [Python](http://www.python.org/) programma chiamato [CWM](http://www.w3.org/2000/10/swap/cwm.py) o "Macchina mondo chiuso". Ulteriori informazioni possono essere trovate sul [SWAP](http://www.w3.org/2000/10/swap/) sito.

Al momento, la migliore dimostrazione del suo uso può essere come si può convertire da RDF XML in Notation3 e viceversa. Per convertire "a.n3" in "a.rdf", è sufficiente utilizzare la seguente riga di comando: -

python cwm.py a.n3 -rdf> a.rdf

CWM è un potente kit di strumenti Web Semantico, e noi saremo riferiti al occasionalmente throught questo articolo.

**Semplice Modellazione dati: Schemata**

Il primo "livello" del Semantic Web sopra la sintassi discusso sopra è il semplice modello datatyping. A "schema" (plurale "schemi") è semplicemente un documento o un pezzo di codice che controlla una serie di termini in un altro documento o parte di codice. E 'come una lista di controllo principale, o la definizione della grammatica.

**RDF Schema**

RDF Schema (anche: [Candidate Recommendation RDF Schema](http://www.w3.org/TR/2000/CR-rdf-schema-20000327/) ) è stato progettato per essere un semplice modello datatyping per RDF. Utilizzando RDF ​​Schema, si può dire che "Fido" è un tipo di "Dog", e che "Dog" è una classe secondaria di animali. Possiamo anche creare proprietà e classi, e anche fare un po 'un po' più roba "avanzate" come la creazione di catene e domini per le proprietà.

Tutti i termini per RDF Schema Iniziamo con "http://www.w3.org/2000/01/rdf-schema#", che avrete notato è nella nostra lista di alias "standard" di cui sopra. L'alias "rdfs:" è spesso usato per RDF Schema, e noi continuiamo questa tradizione qui.

I primi tre concetti più importanti che RDF e RDF Schema ci danno sono le "risorse" (rdfs: Resource), la "classe" (rdfs: Class), e la "proprietà" (rdf: Property). Questi sono tutti "classi", in che termini possono appartenere a queste classi. Ad esempio, tutti i termini di RDF sono tipi di risorsa. Dichiarare che qualcosa è un "tipo" di qualcos'altro, usiamo il rdf: tipo di proprietà: -

rdfs: Resource rdf: Tipo rdfs: Class.

rdfs: Class rdf: Tipo rdfs: Class.

rdf: RDF proprietà: Tipo rdfs: Class.

rdf: Tipo rdf: Tipo rdf: Property.

Questo dice semplicemente che "la risorsa è un tipo di classe, classe è un tipo di classe, di proprietà è un tipo di classe, e il tipo è un tipo di proprietà". Queste sono tutte affermazioni vere.

E 'abbastanza facile per compensare le proprie classi. Ad esempio, creiamo una classe denominata "Dog", che contiene tutti i cani del mondo: -

: Dog rdf: Tipo rdfs: Class.

Ora possiamo dire che "Fido è un tipo di cane": -

: Fido rdf: Tipo: Dog.

Possiamo anche creare proprietà abbastanza facilmente dicendo che un termine è un tipo di rdf: Property, e quindi utilizzare queste proprietà nel nostro RDF: -

: Nome rdf: Tipo rdf: Property.

: Fido: nome di "Fido".

Perché abbiamo detto che il nome di Fido è "Fido"? Poiché il termine ": Fido" è un URI, e abbiamo potuto facilmente potuto scegliere qualsiasi URI per Fido, tra cui ": Squiggle" o ": n508s0srh". Abbiamo appena accaduto per utilizzare l'URI ": Fido" perché è più facile da ricordare. Tuttavia, dobbiamo ancora dire macchine che il suo nome è Fido, perché, anche se la gente può intuire che dalla notazione URI (anche se probabilmente non dovrebbe), le macchine non possono.

RDF Schema ha anche un paio di altre proprietà che possiamo fare uso di: rdfs: subClassOf e rdfs: subPropertyOf. Questi ci permettono di dire che una classe o di proprietà è una classe secondaria o di proprietà di un altro sub. Ad esempio, si potrebbe voler dire che la classe "Dog" è una classe secondaria della classe "Animal". Per fare questo, diciamo semplicemente: -

: Dog rdfs: subClassOf: Animal.

Quindi, quando si dice che Fido è un cane, stiamo anche dicendo che Fido è un animale. Possiamo anche dire che ci sono altre sotto classi di animali: -

: Rdfs umane: subClassOf: Animal.

: Rdfs Anatra: subClassOf: Animal.

E poi creare nuove istanze di quelle classi: -

: Bob rdf: Tipo: Human.

: Quakcy rdf: Tipo: Duck.

E poi siamo in grado di inventare un altro immobile, l'uso che, e costruire ancora più informazioni

: Possiede rdf: Tipo rdf: Property.

: Bob: possiede: Fido.

: Bob: possiede: Quacky.

: Bob: nome di "Bob Fleming".

: Quacky: nome "Quakcy".

E così via. Si può vedere che RDF Schema è molto semplice, e tuttavia permette di costruire su basi di conoscenza dei dati in RDF molto molto rapidamente.

I prossimi concetti che RDF Schema ci fornisce, che sono importanti per ricordare, sono le gamme e domini. Intervalli e domini diciamo quali classi soggetto e oggetto di ogni proprietà deve appartenere. Ad esempio, si potrebbe voler dire che la proprietà ": Titolo del libro" deve sempre applicarsi a un libro, e hanno un valore letterale: -

: Libro rdf: Tipo rdfs: Class.

: Titolo del libro rdf: Tipo rdf: Property.

: Titolo del libro rdfs: domain: Libro.

: Titolo del libro rdfs: rdfs gamma: letterali.

: MyBook rdf: Tipo: Libro.

: MyBook: Titolo del libro "My Book".

rdfs: dominio dice sempre quale classe oggetto di una tripla con la proprietà appartiene, e rdfs: gamma dice sempre quale classe oggetto di una tripla con la proprietà appartiene.

RDF Schema contiene anche una serie di proprietà di annotare schemi, offrendo commenti, etichette e simili. Le due proprietà per fare questo sono rdfs: label e rdfs: commenti, e un esempio del loro utilizzo è: -

: Titolo del libro rdfs: label "Titolo del libro";

rdfs: Commento "il titolo di un libro".

È un buon “best practice” per identificare e commentare le vostre nuove proprietà, classi, e altri termini sempre.