RDF – RDFS – RDF/XML

Dott.sa Vincenza Anna Leano
vincenzaanna.leano@unina.it

Basi di Dati II mod. B
Prof. F. Cutugno
A.A. 2010/2011

Esempio

- Concetto “Il prof Cutugno insegna Basi di Dati II”
- Rappresentazione xml:

```
<Corso nome=“Basi di Dati II”>
        <Insegnante> Cutugno</Insegnante>
</Corso>
```

```
<Insegnante nome=“Cutugno”>
        <Corso> Basi di Dati II</Corso>
</Insegnante>
```

```
<OffertaDidattica>
        <Insegnante> Cutugno</Insegnante>
        <Corso>Basi Di Dati II</Corso>
<OffertaDidattica>
```

RDF: Resource Description Framework

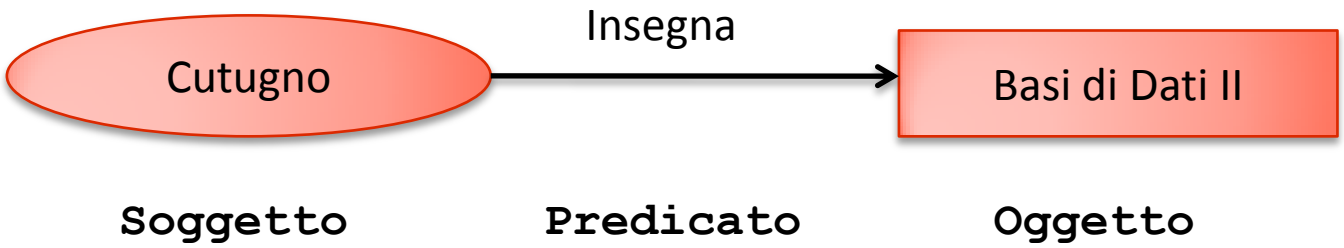
- **RDF** è stato progettato dal W3C per la codifica, lo scambio e il riutilizzo di metadati strutturati
- **RDF è un formato che consente di rappresentare grafi orientati ed etichettati**
- È costituito da due componenti:
  - **RDF Model and Syntax**: espone la struttura del modello RDF, e descrive una possibile sintassi.
  - **RDF Schema**: espone la sintassi per definire schemi e vocabolari per i metadati.

RDF – Data Model and Syntax

Concetti base

RDF Data Model: Concetto base

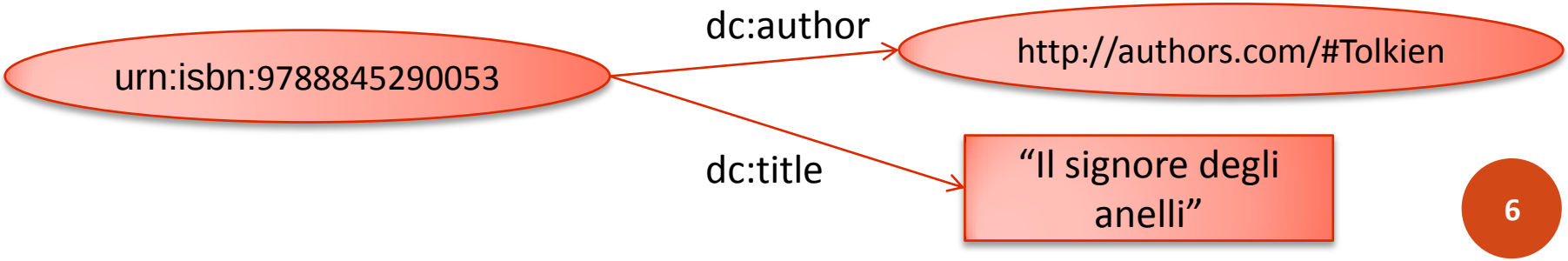
- Triple data-Model: Soggetto – Predicato – Oggetto



- Vincoli:
  - Soggetto e predicato: risorsa
  - Oggetto: Una risorsa o un letterale
- Una tripla <s,P,o> è interpretata come una formula nella logica del primo ordine: P(s,o)
- RDF genera un grafo orientato ed etichettato

Rappresentazione Triple: Grafo

- Una tripla è rappresentata da due noti collegati tramite un arco orientato ed etichettato
  - Soggetto e oggetto: nodi
  - Proprietà: arco
- Rappresentazione
  - Soggetto: Ovale
  - Proprietà: Freccia etichettata
  - Oggetto: ovale se una risorsa, rettangolo se un letterale



Rappresentazione: N-Triple

- Gli URI vengono indicati tra parentesi angolari “<“ e “>”
- I letterali vengono racchiusi tra virgolette
- Dichiarazione dei namespace con @prefix

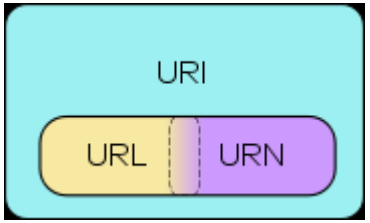
```
@prefix dc: http://dublincore.org/documents/dcmi-namespaces

<urn:isbn:9788845290053> <dc:author> <http://author.org/#Tolkien>
<urn:isbn:9788845290053> <dc:title> “Il signore degli anelli”
```

Risorse

- Una risorsa può essere:
  - Una pagina Web (`http://www.amazon.it`)
  - Una persona (`http://leano.it`)
  - Un libro (`urn:isbn:0-395-36341-1`)
  - Un frammento di codice XML
  - Una proprietà
  - …
- Le risorse sono identificate tramite un URI

URI

- L'Uniform Resource Identifier (URI) è una stringa di caratteri usata per identificare un nome o una risorsa nel web.



- Un URI può essere un URL o un URN:
- Uniform Resource Name (URN) definisce l'identità di un oggetto
  - `urn:isbn:0-395-36341-1`
- Uniform Resource Locator (URL) definisce come accedervi
  - `http://www.unina.it`
- Possiamo usare i namespaces per non dover scrivere sempre gli uri:
  - `namespace unina= http://unina.it`

Letterali

- Possono comparire solo come oggetto nelle triple
- Letterali propri:
  - `"Sono una stringa"`
  - `"Sono una stringa"@it` (dichiarazione della lingua)
- Letterali tipizzati: è possibile aggiungere un tipo al letterale, utilizzando quelli presenti in xsd:
  - `"Sono una stringa"^^xsd:string`
  - `"1"^^xsd:integer`

Blank Nodes (Risorse anonime)

- Un blank node è un oggetto o un soggetto della tripla RDF che non è identificato da un URI e non è un letterale
- Serve a rappresentare elementi sconosciuti
- Vengono interpretati come variabili quantificate esistenzialmente
- Sintassi:
  - `_:p` dove p è un nome locale
- Esempio
  - Marco conosce qualcuno che ha scritto il libro “Introduzione a Java”

```
<myuri:Marco> <myuri:Knows> _:p
_:p <myuri:wrote> “Introduzione a Java”
```

Reificazione (I)

- La reificazione è il processo tramite il quale si rendono oggetto le asserzioni (statement)
- Utile quando si vogliono fare asserzioni su altre asserzioni: aggiungere meta-informazione alle (meta) informazioni che già si hanno.
- E' simile a quello che si fa con gli ER per tradurre in uno schema logico le relazioni molti-molti
- L'asserzione è una relazione ternaria che associa soggetto, predicato ed oggetto
- Posso reificarla costruendo una risorsa che la identifichi e rendendo la relazione ternaria tramite tre relazioni binarie

Reificazione (II)

- Un asserzione viene resa oggetto tramite i costrutti:
  - rdf:statement, rdf:subject, rdf:predicate, rdf:object.
- Esempio: "Maria afferma che l'autore del libro è Marco"
- Fase 1: rendere oggetto l'asserzione "l'autore del libro è Marco"

```
<myuri:#book1> <dc:author> "Marco"
```

```
<#myStatement> <rdf:type> <rdf:Statement>
<#myStatement> <rdf:subject> <myuri:#book1>
<#myStatement> <rdf:predicate> <dc:author>
<#myStatement> <rdf:object> "Marco"
```

Reificazione (II)

- Un asserzione viene resa oggetto tramite i costrutti:
  - rdf:statement, rdf:subject, rdf:predicate, rdf:object.
- Esempio: “Maria afferma che l’autore del libro è Marco”
- Fase 2: Aggiungere la meta informazione

```
<myuri:#Maria> <myuri:claims> <#myStatement>
```

RDFS: RDF Schema

RDFS

- Usato per esprimere vincoli sulle risorse e sulle proprietà
- Permette di dichiarare oggetti e soggetti come istanze di una classe
- Consente di esprimere vincolare dominio e codominio di una proprietà
- Esprime le relazioni semantiche tra le risorse e le proprietà
- Ogni vincolo RDFS viene interpretato come un quantificatore Universale
- I vincoli vengono espressi come triple RDF

RDFS – Classi

- Dichiarazione di Classe

```
<myuri:#Student> <rdf:type> <#rdfs:Class>
```

- Dichiarazione di istanze di una classe

```
<myuri:#Paola> <rdf:type> <myuri:#Student>
```

- Creazione di gerarchie

```
<#Student> <rdfs:subClassOf> <#Person>
```

Proprietà

- Dichiarazione di proprietà:

```
<myuri:#hasName> <rdf:type> <rdf:Property>
```

- Gerarchia di proprietà:

```
<myuri:#hasMother> <rdfs:subPropertyOf>
<myuri:#hasParent>
```

Dominio e codominio

- Restrizione di dominio:
  - “La proprietà #hasName si applica solo alla classe #Person”

```
<myuri:#hasName> <rdfs:domain> <myuri:#Person>
```

- Restrizione di Codominio:
  - “Il tipo della proprietà #hasName è #xsd:string”

```
<myuri:#hasName> <rdfs:range> <xsd:string>
```

Interpretazione in logica del primo ordine

Tripla RDF	Interpretazine in FOL
i rdf:type C	C(i)
i P j	P(i, j)
C rdfs:subClassOf D	∀X (C(X)⇒D(X))
P rdfs:subPropertyOf R	∀X ∀Y (P(X,Y)⇒R(X,Y))
P rdfs:domain C	∀X ∀Y (P(X,Y)⇒C(X))
P rdfs:range D	∀X ∀Y (P(X,Y)⇒D(Y))

RDF/XML

Serializzazione di RDF

Rappresentazione: RDF/XML

- Grafi e N-triple non sono adatti allo scambio di informazione
- Vantaggi di una rappresentazione XML:
  - Serializzazione
  - Definizione di Schema
  - Standard
  - Gestione codifiche
  - Strumenti di navigazione/Query
  - Possibilità di trasformazione tra diversi formati (XSLT)

RDF XML Document (I)

```
<?xml version="1.0"?>
```

- Prologo xml

RDF XML Document (II)

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
  xmlns:myuri= "http://myuri.com"
  xmlnd:dc="http://dublincore.org/documents/dcmi-
namespaces"



</rdf:RDF>
```

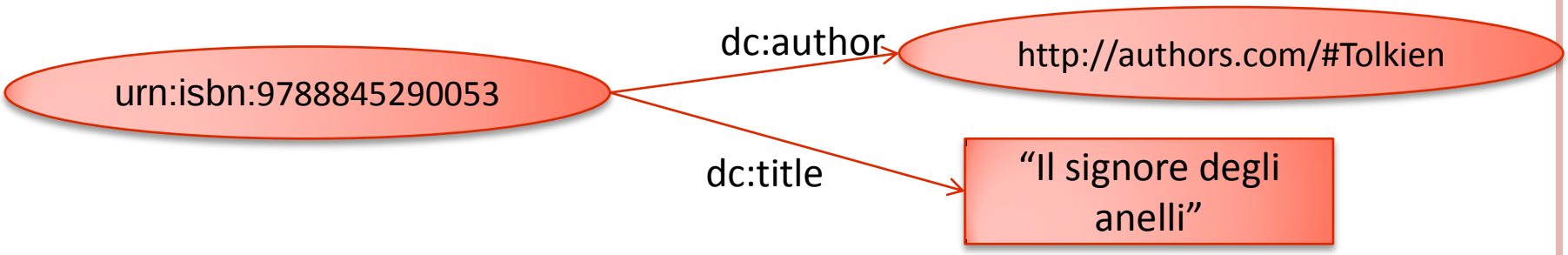
- Root element <rdf:RDF> con dichiarazione dei namespace per gli uri

RDF XML Document (III)

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
  xmlns:myuri= "http://myuri.com"
  xmlnd:dc=": http://dublincore.org/documents/dcmi-
namespaces"
 <rdf:Description rdf:about="unr:isbn:9788845290053">
       <dc:author>Tolkien</dc:author>
 </rdf:Description>
</rdf:RDF>
```

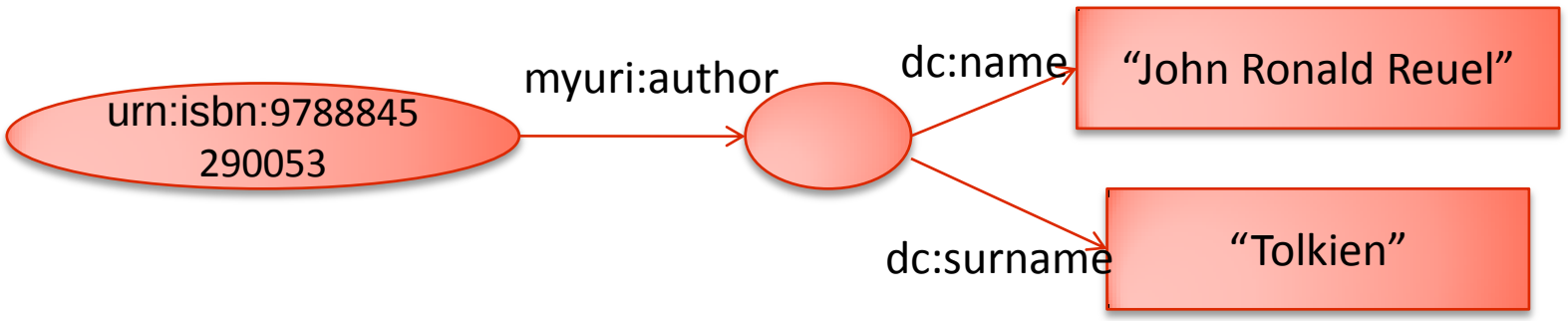
- Collezione di elementi Description:
  - rdf:about= soggetto
  - Tag figlio= proprietà
  - Valore tag figlio= oggetto

Accorpamento di proprietà




```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlnd:dc=": http://dublincore.org/documents/dcmi-namespaces"
 <rdf:Description rdf:about="unr:isbn:9788845290053">
       <dc:title> Il signore degli Anelli </dc:title>
       <dc:auhor> http://authors.com/#Tolkien</dc:author>
 </rdf:Description>
</rdf:RDF>
```

Proprietà Complesse




```
<rdf:Description rdf:about="unr:isbn:9788845290053">
        <myuri:author>
            <dc:name> John Ronald Reuel </dc:name>
           <dc:surname> Tolkien</dc:surname>
        </myuri:author>
 </rdf:Description>
```

RIFERIMENTO A RISORSE INTERNE AL DOCUMENTO

```
<rdf:Description rdf:about="myuri:book001">
  <dc:author rdf:resource="myuri:Tolkien"/>
</rdf:Description>

<rdf:Description rdf:about="myuri:Tolkien">
       <dc:surname>Tolikien</dc:surname>
</rdf:Description>
```

```
<rdf:Description rdf:about="myuri:book001">
  <dc:author>
    <rdf:Description rdf:about="myuri:Tolkien">
       <dc:surname>Tolkien</dc:surname>
    </rdf:Description>
  </dc:author>
</rdf:Description>
```

Abbreviazione di oggetti letterali

- La proprietà diventa un attributo del tag rdf:Description che ha come valore il letterale

```
<rdf:Description rdf:about="unr:isbn:9788845290053">
        <dc:title> Il signore degli Anelli </dc:title>
</rdf:Description>
```

```
<rdf:Description rdf:about="unr:isbn:9788845290053"
dc:title="Il signore degli Anelli"/>
```

Abbreviazione di oggetti risorse

- E' possibile eliminare le description innestate indicando nel predicato che l'oggetto deve essere considerato una risorsa

```
<rdf:Description rdf:about="myuri:book001">
  <dc:author>
    <rdf:Description rdf:about="myuri:Tolkien">
      <dc:surname>Tolkien</dc:surname>
    </rdf:Description>
  </dc:author>
</rdf:Description>
```

```
<rdf:Description rdf:about="myuri:book001">
  <dc:author rdf:parseType="Resource">
      <dc:surname>Tolkien</dc:surname>
  </dc:author>
</rdf:Description>
```

Blank Node (I)

- Tramite l'attributo rdf:id possiamo rappresentare le risorse anonime
- ES: "Marco conosce qualcuno che ha scritto il libro "Introduzione a JAVA"
  - `<myuri:Marco> <myuri:Knows> _:p`
  - `_:p <myuri:wrote> "Introduzione a Java"`

```
<rdf:Description rdf:about="myuri:Marco">
  <myuri:Knows rdf:ID="#p01"/>
</rdf:Description>

<rdf:Description rdf:ID="p01">
 <myuri:wrote>Introduzione a JAVA</myuri:wrote>
</rdf:Description>
```

Blank Node (II)

- Possiamo ommettere il tag description per il blank node, utilizzando l'attributo rdf:parseType

```
<rdf:Description rdf:about="myuri:Marco">
  <myuri:Knows rdf:parseType="Resource">
   <myuri:wrote>Introduzione a JAVA</myuri:wrote>
  </myuri:Knows>
</rdf:Description>
```

TIPIZZAZIONE

```
<rdf:Description rdf:about="myuri:Tolkien">
  <dc:surname
rdf:dataType="http://www.w3.org/2001/XMLSchema#string
"> Tolikien </dc:surname>
</rdf:Description>
```

CONTAINERS

- Per raggruppare gli elementi RDF mette a disposizione tipi e proprietà per descrivere dei gruppi.
- Il **`Container`** è una risorsa che raggruppa degli elementi. Gli elementi di un **`Container`** vengono chiamati **`Member`**. I **`Member`** possono essere risorse o letterali.
- Tre tipi di **`Container`** :
  - **`rdf:Bag`**: gruppo di elementi in cui non importa l'ordine e possono esistere duplicati
  - **`rdf:Seq`**: gruppo di elementi dove l'ordine è significativo e possono esistere duplicati
  - **`rdf:Alt`**: gruppo di elementi che rappresentano delle alternative (esempio lista di mirror da cui è possibile scaricare un p
- Gli elementi di un container possono essere racchiusi da due tipi di tag:
  - **`rdf:_n`**: se è importante la numerazione (rdf:_1, rdf:_2..)
  - **`rdf:li`**: se non importa numerarli

Bag

- Elenco di studenti che seguono il corso “Basi di Dati II”

```
<rdf:Description
rdf:about="http://informatica.dsf.unina.it/BDD2">
  <myuri:students>
    <rdf:Bag>
      <rdf:li rdf:resource="myuri:student01"/>
      <rdf:li rdf:resource="myuri:student02"/>
      <rdf:li rdf:resource="myuri:student03"/>
    </rdf:Bag>
  </myuri:students>
<rdf:Description>
```

Seq

- Vincitori di un concorso

```
<rdf:Description rdf:about="http://concorso.it">
 <myuri:vincitori>
   <rdf:Seq>
    <myuri:candidato> Tizio </myuri:candidato>
    <myuri:candidato> Caio </myuri:candidato>
    <myuri:candidato> Raccomandato
</myuri:candidato>
   </rdf:Seq>
  </myuri:vincitori>
</rdf:Description>
```

ALT

- Elenco di mirror da cui è possibile scaricare un pacchetto software

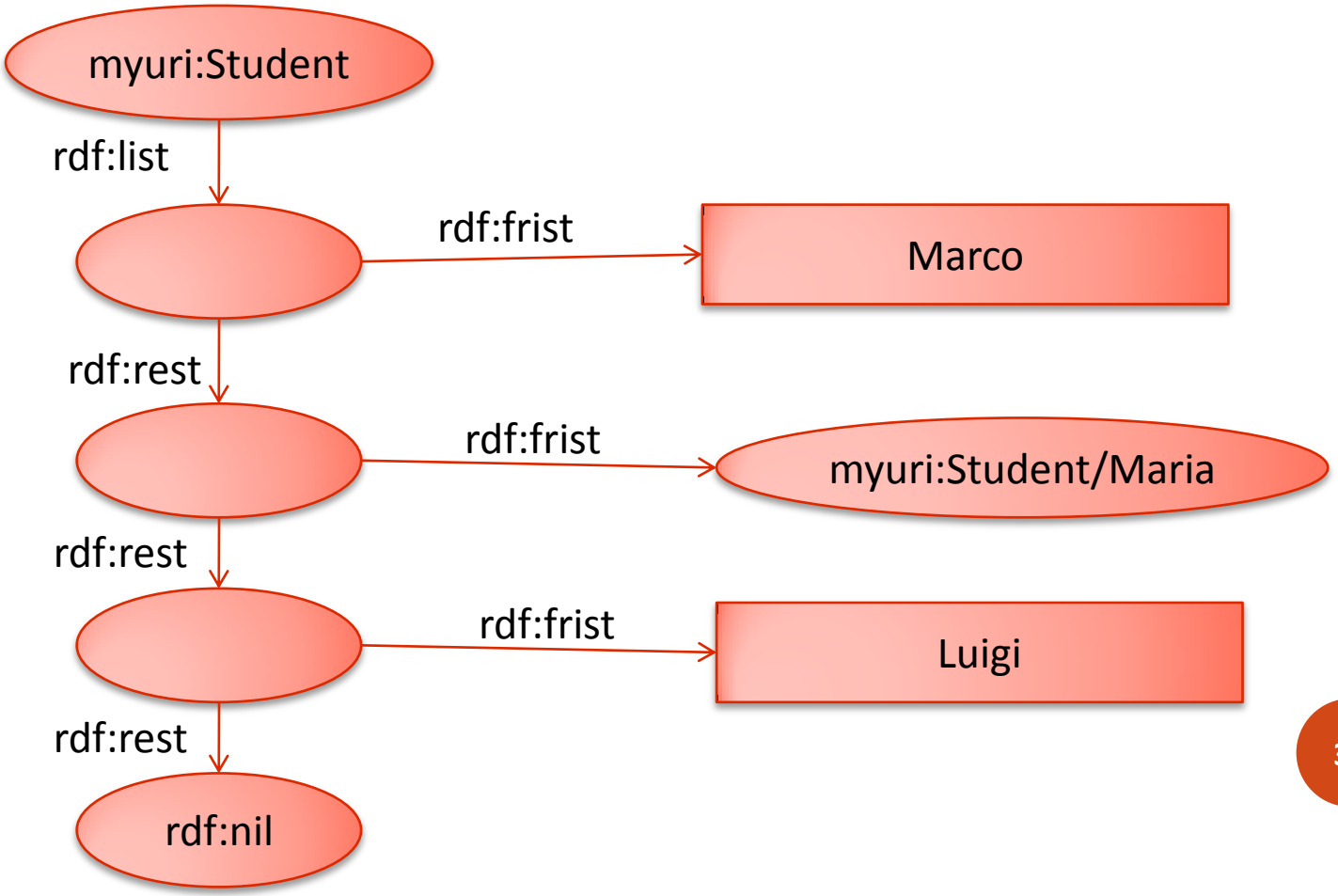
```
<rdf:Description
rdf:about="http://softwarefree.com/a.exe">
  <myuri:mirrors>
    <rdf:Alt>
      <rdf:_1 rdf:resource="http//a.it/a.exe>
      <rdf:_2 rdf:resource="ftp://a.com/a.exe"/>
    </rdf:Alt>
  </myuri:mirrors>
<rdf:Description>
```

Collection

- I **`Container`** non sono gruppi chiusi, possiamo dire che gli oggetti che appartengono al Container godono di quella proprietà, ma nulla al riguardo di quelli che non compaiono
- Per avere un insieme di oggetti “chiuso” si usano le **`Collection`**
- Struttura ricorsiva
  - **`rdf:list, rdf:frist, rdf:rest, rdf:nil`**

Esempio Collection

- Gli studenti di basi 2 sono solo “Marco” , “Maria” e “Luigi”



Esempio Collection - XML

```
<rdf:Description
rdf:about="http://informatica.dsf.unina.it/BDD2">
    <s:students rdf:nodeID="L1"/>
</rdf:Description>
<rdf:Description rdf:nodeID="L1">
    <rdf:first  myuri:Student="Marco"/>
    <rdf:rest rdf:nodeID="L2"/>
</rdf:Description>
<rdf:Description rdf:nodeID="L2">
     <rdf:first rdf:resource="myuri:Stundent/Maria"/>
     <rdf:rest rdf:nodeID="L3"/>
 </rdf:Description>
<rdf:Description rdf:nodeID="L3">
     <rdf:first>
        <myuri:student>Luigi</myuri:student>
     </rdf:first>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#nil"/>
</rdf:Description>
```

Riferimenti Bibliografici

- Abiteboul - Web Data Management and Distribution– Cap 8
- Shelley Powers - Practical RDF - O'Reilly Media
- http://www.w3.org/TR/rdf-primer/
- http://www.w3.org/TR/rdf-concepts/
- http://www.w3.org/TR/rdf-mt/
- http://www.w3.org/TR/rdf-syntax-grammar/