Introduction to RDF and RDFS
Charlie Abela
charlie.abela@um.edu.mt
Lecture Outline

• Introduction to RDF
  – Basic building blocks: resources
  – Different notations for RDF
    • Triples
    • Graphs
    • RDF/XML
  – Exercise

• RDFS: compliments RDF
  – Exercise
Moving up the layers

User Interface & applications

Trust

Proof

Unifying Logic

Query: SPARQL

ontology: OWL

Rules: RIF

RDF-S

Data interchange: RDF

XML

URI

Unicode

Crypto
Scenario

- Jim’s searching experience over the Web is not brilliant
- It gets really frustrating at times when he needs to find relevant information
Scenario (2)

- Jim’s colleague John has recently started using an application called PEA (Personal Assistant).
- PEA can learn about the user’s interests and is able to help out John in many tasks.
- During its learning process PEA implicitly gathers information about John’s interests.
  - John is interested in music.
  - John keeps a favourites’ list of song titles.
Scenario (3)

- After some time observing John’s behaviour PEA has gathered some facts:
  - John has a list of favourite songs
  - List of favourite songs is ranked according to John’s preferences
  - John’s favourite song is “I Feel Good”
  - James Brown sings “I Feel Good”
  - “I Feel Good” is a song
  - A song has lyrics
  - John is an instance of a Person
  - James Brown is an instance of a Person
  - A video of the song “I Feel Good” is found at http://www.youtube.com/watch?v=IWcNiebYGuo
RDF Introduction

• Resource Description Framework is a language whose main intent is to provide a common framework for expressing machine-processable information.

• Through RDF it is possible to represent both physically accessible entities, as well as abstract entities (i.e. those that cannot be retrieved):
  – physical entity: a person’s Web page
  – abstract entity: the resource pointing to information about the person himself

• It also facilitates information exchange between different applications.
RDF: The Basics

• Basic building block: `<subject predicate object>` triple
  – It is called a **statement**
  – Sentence:
    • “*Dan Brown is the author of the Da Vinci Code*” is such a statement
      – Subject: *Dan Brown*
      – Predicate: *authorOf*
      – Object: *Da Vinci Code*

• RDF has been given an XML syntax
  – This syntax inherits the benefits of XML
RDF: The Basics (2)

- The fundamental components of RDF are:
  - Resources: anything defined through a URI
    - http://www.category.com/authors#Dan_Brown
  - Properties: resources that describe a relation
    - books:authorOf (authorOf relation defined in books domain)
    - essentially binary relations
  - Statements: assign a value to a property associated with a specific resource
    - <#auth1102 books:authorOf “Da Vinci Code”>
Resources

• We can think of a resource as an object, a “thing” we want to talk about (i.e. has an identity)
  – E.g. authors, books, publishers, places, people

• Every resource (physical or abstract) has a URI, a Universal Resource Identifier

• A URI can be
  – a URL (Web address)
    • http://www.bookresource.com/pub/books.html
  – some other kind of unique identifier
    • #book102 (references a unique book resource)
Properties

- Properties are special kinds of resources
- They describe relations between resources
  - E.g. “written_by”, “age”, “title”, etc.
- Properties are also identified by URIs
  - authorOf relation used in association with a book resource

```xml
<books:authorOf>Da_Vinci_Code</books:authorOf>
<books:authorOf rdf:resource="#book102"/>
```
Statements

• RDF statements assert the properties of resources
• Values can be **resources** or **literals**
  – Literals are atomic values (strings)
• Three views of an RDF statement
  – as a triple
  – as a directed graph
  – as RDF/XML serialisation
Example: As a Triple

• The triple \((x, P, y)\) can be considered as a logical formula \(P(x, y)\)
  – Binary predicate \(P\) relates object \(x\) to object \(y\)
  – RDF offers only binary predicates (properties)
  – Any n-ary relation in RDF has to be converted into a set of binary relations
Example: As a Graph

• A directed graph with labeled nodes and arcs
  – from the resource (the subject of the statement)
  – to the value (the object of the statement)
• Known in AI as a semantic net

http://www.bookstore.com/books#auth1102

http://www.authorsInfo.com/authors#name  http://www.booksInfo.com/books#authorOf

"Dan Brown"  "Da Vinci Code"
An RDF document is represented by an XML element with the tag `rdf:RDF`.

The content of this element is a number of descriptions, which use `rdf:Description` tags.

Every description makes a statement about a resource, usually identified in 2 ways:
- an `about` attribute, referencing an existing description
- an `ID` attribute, creating a new description

```
<rdf:Description rdf:about="#auth1102">
```
RDF/XML (2)

- The `rdf:Description` element makes a statement about the resource `auth1102`.
- Within the description:
  - the property is used as a tag
  - the content is the value of the property

```
<authors:name>Dan Brown</authors:name>
```
Example: RDF/XML

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:books="http://www.booksInfo.com/books#"
   xmlns:authors="http://www.authorsInfo.org/authors#">

   <rdf:Description rdf:ID="auth1102">
      <authors:name>Dan Brown</authors:name>
      <books:authorOf>Da_Vinci_Code</books:authorOf>
   </rdf:Description>

</rdf:RDF>
```
Concise representation

<!DOCTYPE rdf:RDF [  
<!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">  
<!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#">  
<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">  
<!ENTITY books "http://www.booksInfo.org/books#">  
<!ENTITY authors "http://www.authorsInfo.org/authors#">]>

<rdf:RDF  
xmlns:rdf ="&rdf;"  
xmlns:rdfs="&rdfs;"  
xmlns:xsd="&xsd;"  
xmlns:authors="&authors;"  
xmlns:books="&books;">

  <rdf:Description rdf:ID="auth1102">  
    <authors:name>DanBrown</authors:name>  
    <books:authorOf rdf:resource="#book102"/>
  </rdf:Description>

</rdf:RDF>
XML Schema data types

• In RDF, typed literals can be used
  – Usually through the use of XML Schema
• In which case the XML Schema namespace has to be declared in the namespace block

<rdf:Description rdf:ID="auth1102">
  <authors:name rdf:datatype="&xsd;string">Dan Brown</authors:name>
  <books:authorOf rdf:resource="#book102"/>
  <authors:dob rdf:datatype="&xsd;string">22/06/1964</authors:dob>
</rdf:Description>
Using `rdf:type`

- Declare a statement which formally defines a description

```xml
<rdf:Description rdf:ID="auth1102">
  <rdf:type rdf:resource="&authors;author"/>
  <authors:name rdf:datatype="&xsd;string">Dan Brown</authors:name>
  <books:authorOf rdf:resource="#book102"/>
  <authors:dob rdf:datatype="&xsd;string">22/06/1964</authors:dob>
</rdf:Description>

- Declares `auth1102` to be of type `author` (itself defined in the authors vocabulary)
<rdf:RDF
   xmlns:rdf="&rdf;"
   xmlns:xsd="&xsd;"
   xmlns:books="&books;"
   xmlns:authors="&authors;">

   <rdf:Description rdf:ID="book102">
       <books:title rdf:datatype="&xsd;string">Da Vinci Code</books:title>
   </rdf:Description>

   <rdf:Description rdf:ID="auth1102">
       <rdf:type rdf:resource="&authors;author"/>
       <authors:name rdf:datatype="&xsd;string">Dan Brown</authors:name>
       <books:authorOf rdf:resource="#book102"/>
       <authors:dob rdf:datatype="&xsd;string">22/06/1964</authors:dob>
   </rdf:Description>

</rdf:RDF>
XML Vs RDF

```xml
<book ID="book102">
  <author ID="auth1102" name="Dan Brown"/>
  <title name="Da Vinci Code"/>
</book>
```

```xmlox
<rdf:Description rdf:ID="book102">
  <books:title>Da Vinci Code</books:title>
</rdf:Description>
```

```
xmlo XML
```

```xmlox
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf rdf:resource="#book102"/>
</rdf:Description>
```

```xmlox RDF/XML
```

```
xmlox RDF
```
XML vs RDF II

[Diagram showing the structure of an RDF graph, with nodes for book, ID, author, title, name, and ISBN, and values for auth1102, Dan Brown, Da Vinci Code, and 978-0385504201.]
XML vs RDF III

http://www.bookstore.com/books#book102

http://www.booksInfo.com/books#title

978-0385504201

http://www.authorsInfo.org/authors#authorOf

http://www.authorsInfo.org/authors#name

22/06/64

Dan Brown

http://www.authorsInfo.org/authors#dob

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

authors:author

http://www.bookstore.com/books#auth1102


Da Vinci Code
Container Elements

- Collections of resources or attributes can be defined within a **container** element.
- RDF provides three types of containers:
  - **rdf:Bag** an unordered container, may contain multiple occurrences
  - **rdf:Seq** an order container, may contain multiple occurrences
  - **rdf:Alt** a set of alternatives
Container Elements: Bag

• Dan Brown is the author of three books (defined elsewhere), order of publication is not important

```xml
<rdf:Description rdf:ID="auth1102">
    <authors:name>Dan Brown</authors:name>
    <books:authorOf>
        <rdf:Bag>
            <rdf:li rdf:resource="#book106"/>
            <rdf:li rdf:resource="#book102"/>
            <rdf:li rdf:resource="#book103"/>
        </rdf:Bag>
    </books:authorOf>
</rdf:Description>
```
Container Elements: Seq

• However if it's needed to specify that `authorOf` specifies ordering information (e.g. based on publication dates)

```xml
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf>
    <rdf:Seq>
      <rdf:li rdf:resource="#book102"/>
      <rdf:li rdf:resource="#book103"/>
      <rdf:li rdf:resource="#book106"/>
    </rdf:Seq>
  </books:authorOf>
</rdf:Description>
```
Container Elements: Alt

• *Alt* implies a list of alternatives from which only one can be chosen

```xml
<rdf:Description rdf:ID="book102">
  <books:title>Da Vinci Code</books:title>
  <book:language>
    <rdf:Alt>
      <rdf:li>engdoc</rdf:li>
      <rdf:li>mtdoc</rdf:li>
    </rdf:Alt>
  </book:language>
</rdf:Description>
```
Closed container

• If we need to specify that the listed resources are **ALL** the members of the container we need to use the **List** structure

```xml
<books:authorOf>
  <rdf:List>
    <rdf:first>
      <rdf:Description rdf:about="#book103"/>
    </rdf:first>
    <rdf:rest>
      <rdf:List>
        <rdf:first>
          <rdf:Description rdf:about="#book102"/>
        </rdf:first>
        <rdf:rest>
          <rdf:Description rdf:about="&rdf;nil"/>
        </rdf:rest>
      </rdf:List>
    </rdf:rest>
  </rdf:List>
</books:authorOf>
```
Closed collection

• A shorthand syntax for a closed collection exists

```xml
<rdf:Description rdf:ID="auth1102">
  <authors:name>Dan Brown</authors:name>
  <books:authorOf rdf:parseType="Collection">
    <rdf:Description rdf:about="#book103"/>
    <rdf:Description rdf:about="#book102"/>
  </books:authorOf>
</rdf:Description>
```
Describing Bookmarks

```xml
<rdf:Description rdf:nodeID="bk001">
  <rdf:type rdf:resource="&bookmarks;bookmark"/>
  <bookmarks:url rdf:resource="http://news.google.com/"/>
  <bookmarks:date rdf:datatype="&xsd;date">
    12/12/07</bookmarks:date>
  <bookmarks:query rdf:datatype="&xsd;string">
    news + google</bookmarks:query>
  <bookmarks:title rdf:datatype="&xsd;string">
    Google News</bookmarks:title>
  <bookmarks:category rdf:datatype="&xsd;string">
    News</bookmarks:category>
</rdf:Description>

.rdf:nodeID is used to indicate an anonymous node
```
Describing a Process

```xml
<rdf:Description rdf:nodeID="proces001">
  <rdf:type rdf:resource="&process;sysprocess"/>
  <process:name rdf:datatype="&xsd;string">
    lsass</process:name>
  <process:ext rdf:datatype="&xsd;string">
    exe</process:ext>
  <process:folder>system32</process:folder>
  <process:version>5.1.2600.2180</process:version>
  <process:company>MicrosoftCorporation</process:company>
</rdf:Description>
```
Exercises

• Write the following statements in RDF and draw the associated graphs (use any required namespaces)
  – The library is located in Bremen
  – The author William Start wrote the books with titles “Modern Web Services” and “Theory of Computation”
  – The email address of lecturer John Brown is john.brown@widget.org
  – Lecturer John Brown has an office in the Department of AI. The office is situated on the 2nd floor of the Computer Building.

• Write RDF statements about a catalogue of products.
  – Each catalogue has an ID and a description and consists of a list of products
  – Each product is associated with a department and has a product ID and a description
  – Multiple products may be associated with the same department
RDFS
• The library is located in Bremen

```xml
<rdf:Description rdf:about="library">
  <books:libraryLoc rdf:datatype="&xsd;string">Bremen</books:libraryLoc>
</rdf:Description>
```
The book with title “Artificial Intelligence” and ISBN 978-123458 is written by the author Henry Wise

```
<rdf:Description rdf:about="book012">
  <rdf:type rdf:resource="&books;Book"/>
  <books:title rdf:datatype="&xsd;string">
    Artificial Intelligence</books:title>
  <books:isbn rdf:datatype="&xsd;string">
    ISBN 978123478</books:isbn>
  <books:author rdf:resource="#author12"/>
</rdf:Description>

<rdf:Description rdf:about="author12">
  <books:authorName rdf:datatype="&xsd;string">
    Henry Wise</books:authorName>
</rdf:Description>
```
The author William Start wrote the books with titles “Modern Web Services” and “Theory of Computation”
RDF/XML Definition

• The author William Start wrote the books with titles “Modern Web Services” and “Theory of Computation”

<rdf:Description rdf:about="author13">
  <rdf:type rdf:resource="&books;author"/>
  <books:authorName rdf:datatype="&xsd;string">
    William Start
  </books:authorName>
  <books:authorOf rdf:parseType="Collection">
    <rdf:Description rdf:about="#book103"/>
    <rdf:Description rdf:about="#book102"/>
  </books:authorOf>
</rdf:Description>
Moving up the layers
RDF Schema

• Consider the following XML

```xml
<author>Dan Brown</author>
<journalist>John News</journalist>
<magazine title="Internet Computing">
  <contributor>John Davies</contributor>
</magazine>
```

• Though XML is correct, it is semantically unsatisfactory (John News, John Davies and Dan Brown are types of authors)

• Need to be able to define something of this sort:
  – journalists and magazine contributors are all authors
  – authors are writers
RDF Schema II

- RDF makes no assumptions about any specific domain, nor does it define the semantics of such a domain.
- To define the **terminology** related to any domain a schema language such as RDF Schema can be used.
- **Relation** between RDF and RDF Schema is not the same as that between XML and XML Schema:
  - XML schema constraints the structure of XML documents.
  - RDFS defines the vocabulary used within an RDF data model.
  - Through RDFS it's possible to:
    - specify the properties that can be applied to an object.
    - specify the values that such properties can take.
    - declare relationships between objects.
RDF Schema III

• The user can describe any particular domain using:
  – **Classes** and **Properties**: differentiate between individual objects (or instances) and classes of object
    – author: Dan Brown and class of authors
    – restrict properties to apply to certain things
      – *Only authors can contribute to certain magazines* i.e. restrict the domain of contributors to a magazine as ranging only over authors
  – **Class Hierarchies and Inheritance**
    – Classes of authors, writers, journalists, reporters, editors etc
      – authors, journalists and reporters are all writers (journalist is a subclass of writer)
  – **Property Hierarchies**
    – E.g. the property “authorOf” is a subproperty of “contributesTo”
    – However the converse may not necessarily be always true
Core Classes

- `rdfs:Resource`, the class of all resources
- `rdfs:Class`, the class of all classes
- `rdfs:Literal`, the class of all literals (strings)
- `rdf:Property`, the class of all properties.

```xml
<rdfs:Class rdf:about="Process">
  <rdfs:comment>Represents the class of all processes</rdfs:comment>
</rdfs:Class>
```
Core Properties

- **rdf:type**, which relates a resource to its class
  - The resource is declared to be an instance of that class
- **rdfs:subClassOf**, which relates a class to one of its superclasses
  - All instances of a class are instances of its superclass
- **rdfs:subPropertyOf**, relates a property to one of its superproperties

```xml
<rdfs:Class rdf:ID="SysProcess">
   <rdfs:subClassOf rdf:resource="#Process"/>
</rdfs:Class>

<rdf:Property rdf:ID="authorOf">
   <rdfs:subPropertyOf rdf:resource="#contributeTo"/>
</rdf:Property>
```
Core Properties (2)

- **rdfs:domain**, which specifies the domain of a property P
  - The class of those resources that may appear as *subjects* in a triple with predicate P
  - If the domain is not specified, then any resource can be the subject

- **rdfs:range**, which specifies the range of a property P
  - The class of those resources that may appear as *values (of objects)* in a triple with predicate P

```xml
<rdf:Property rdf:ID="procName">
  <rdfs:domain rdf:resource="#Process"/>
  <rdfs:range rdf:range="&xsd:string"/>
</rdf:Property>
```
Example

Diagram:

- Literal
  - has_name
  - has_name

- Contributor
  - subClassOf
  - authorOf
  - subClassOf

- Writer
  - subClassOf
  - subClassOf

- Author

- Book
  - has_name
  - has_editor

- Editor
  - subClassOf

- Journalist
  - subClassOf
<rdfs:Class rdf:ID="Writer"/>

<rdfs:Class rdf:ID="Author">
    <rdfs:comment>all authors that contribute to books</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Writer"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Book">
    <rdfs:comment>The class of books</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="authorOf">
    <rdfs:subPropertyOf rdf:resource="#contributeTo"/>
    <rdfs:domain rdf:resource="#Writer"/>
    <rdfs:range rdf:resource="#Book"/>
</rdf:Property>
RDFS Example II

```xml
<RDFS:Class rdf:about="#Author"/>
  <RDFS:subClassOf rdf:resource="#Writer"/>
</RDFS:Class>

<RDF:Property rdf:ID="phone">
  <RDFS:domain rdf:resource="#Writer"/>
  <RDFS:range rdf:resource="&xsd:Literal"/>
</RDF:Property>
```
Complete Example

```xml
<rdf:RDF
    xmlns:rdf="&rdf;"
    xmlns:xsd="&xsd;"
    xmlns:rdfs="&rdfs;">
    <rdfs:Class rdf:ID="Contributor"/>
    <rdfs:Class rdf:ID="Writer">
        <rdfs:subClassOf rdf:resource="#Contributor"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Author">
        <rdfs:subClassOf rdf:resource="#Writer"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Editor">
        <rdfs:subClassOf rdf:resource="#Contributor"/>
    </rdfs:Class>
    <rdfs:Class rdf:ID="Book"/>
    <rdfs:Property rdf:ID="has_editor">
        <rdfs:domain rdf:resource="#Book"/>
        <rdfs:range rdf:resource="#Editor"/>
    </rdfs:Property>
    <rdfs:Property rdf:ID="has_author">
        <rdfs:domain rdf:resource="#Book"/>
        <rdfs:range rdf:resource="&rdf;Seq"/>
    </rdfs:Property>
    <rdfs:Property rdf:ID="has_name">
        <rdfs:domain rdf:resource="#Book"/>
        <rdfs:domain rdf:resource="#Contributor"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </rdfs:Property>
</rdf:RDF>
```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.booksInfo.com/books#">
  <Book rdf:ID="sw_1020">
    <has_name>Semantic Web Primer</has_name>
    <has_editor>
      <Editor rdf:ID="mit">
        <has_name>MIT</has_name>
      </Editor>
    </has_editor>
    <has_author>
      <rdf:Seq>
        <rdf:li>
          <Author rdf:ID="ga">
            <has_name>Grigoris Antoniou</has_name>
          </Author>
        </rdf:li>
        <rdf:li>
          <Author rdf:ID="fvh">
            <has_name>Frank van Harmelen</has_name>
          </Author>
        </rdf:li>
      </rdf:Seq>
    </has_author>
  </Book>
</rdf:RDF>
Applications of RDF/S: FOAF

- FOAF (Friend of a Friend)
- FOAF is a way of providing affiliation and other social information about yourself
- Also a way of describing a network of friends and others we know, in such a way that automated processes such as web bots can find this information and incorporate it with other FOAF files
- The FOAF-a-matic is a web form and the submitted information is used to generate the RDF/XML for a specific FOAF file and add it to the FOAF network.
<foaf:Person rdf:ID="jhendler">
  <foaf:mbox_sha1sum>0b62d4242736e64be6138547c79a811b3e82fd52</foaf:mbox_sha1sum>
  <foaf:firstName>Jim</foaf:firstName>
  <foaf:surname>Hendler</foaf:surname>
  <foaf:name>Jim Hendler</foaf:name>
  <foaf:title>Tetherless World Constellation Chair</foaf:title>
  <foaf:homepage rdf:resource="http://trust.mindswap.org/cgi-bin/FilmTrust/foaf.cgi?user=jhendler"/>
  <foaf:workplaceHomepage rdf:resource="http://owl.mindswap.org"/>
  <foaf:nick>jhendler</foaf:nick>
  <foaf:knows>
    <foaf:Person rdf:about="http://trust.mindswap.org/cgi-bin/FilmTrust/foaf.cgi?user=golbeck#golbeck">
      <rdfs:seeAlso rdf:resource="http://trust.mindswap.org/cgi-bin/FilmTrust/foaf.cgi?user=golbeck"/>
      <foaf:nick>golbeck</foaf:nick>
    </foaf:Person>
  </foaf:knows>
  <foaf:knows>
    <foaf:Person rdf:about="http://trust.mindswap.org/cgi-bin/FilmTrust/foaf.cgi?user=finin#finin">
      <rdfs:seeAlso rdf:resource="http://trust.mindswap.org/cgi-bin/FilmTrust/foaf.cgi?user=finin"/>
      <foaf:nick>finin</foaf:nick>
    </foaf:Person>
  </foaf:knows>
</foaf:Person>
Photos of People I Know

<foaf:knows>
</foaf:knows>
<foaf:knows>
  <foaf:Person rdf:about="http://www.PeterBorg/pietru#pietru"/>
</foaf:knows>
RDF Tools

- **RDF Store**: stores RDF models in relational dbs (persistent storage)
  - Jena can be hooked with MySQL, PostgreSQL or Oracle db
- **RDF validator**: validates RDF and presents different views (triples and dg) of result
- **RDF Editors**:
  - Protégé: http://protege.stanford.edu/
  - Oiled: http://oiled.man.ac.uk/
- **RDF visualiser**
  - RDF Viz (visualises graphs)
- **Search for schemas**:
  - SWOOGLE: http://swoogle.umbc.edu/
Further Information

• Semantic Web Primer, Chapter 3
• RDF Primer, E. Miller
  – http://www.w3.org/TR/rdf-primer/
• RDF tutorial examples
• *The Semantic Web: roles of XML and RDF*, Stefan Decker,
• Jena RDF tutorial,
  – http://jena.sourceforge.net/documentation.html
Exercise

- Create an RDFS vocabulary for this scenario
- Create an instance file with an author that has published 3 different publications. You can add relevant properties to the basic concepts