NSWI144 – Linked Data – Lecture 2 – 15 October 2012

RDF

Martin Svoboda

Faculty of Mathematics and Physics
Charles University in Prague

Motivation

- Web of Documents
- Web of Data
  - Linked Data – RDF, RDFa
  - Vocabularies – RDFS, OWL
  - Querying – SPARQL
  - Inference
  - Applications

Outline

- RDF
  - Introduction
  - Statements
  - Data model
  - Containers
  - Serialization

Introduction

- RDF
  - **Resource Description Framework**
  - Language for representing information about resources in the World Wide Web
  - W3C

Introduction

- W3C recommendations
  - **RDF Primer**
    - http://www.w3.org/TR/rdf-primer/
  - **RDF Concepts and Abstract Syntax**
    - http://www.w3.org/TR/rdf-concepts/
  - **RDF/XML Syntax Specification**
    - http://www.w3.org/TR/rdf-syntax-grammar/
  - **RDF Semantics**
    - http://www.w3.org/TR/rdf-mt/

Statements

- Idea: statements about resources
  - **Resources**
    - Anything that is identifiable by a URI reference
      - Usually things identified by standard URLs…
      - … but also things that may not be directly retrievable
  - **Statements**
    - Triples inspired by natural languages
    - **Subject Predicate Object**
      - `http://is.cuni.cz/studium/sis#student358`
        `http://is.cuni.cz/studium/sis#name`
        `"John"`

Statements

- Components of triples
  - **Subject**
    - Describes the thing the statement is about
  - **Predicate**
    - Describes the property or characteristic of the subject
  - **Object**
    - Describes the value of that property

Statements

- Identifiers
  - **URI references**
    - URI with an optional fragment identifier
      - `http://is.cuni.cz/studium/sis#student358`
      - `mailto:svoboda@ksi.mff.cuni.cz`
      - `urn:issn:0167-6423`
    - Unicode characters
  - **Qualified names**
    - Similar idea as prefixes for namespaces in XML
      - `sis: = http://is.cuni.cz/studium/sis#`
      - `sis:student358 sis:name "John"`

Statements

- Domains for triple parts
  - **Identifiers**
    - **URI references**
    - **Blank node identifiers**
      - Special and only locally valid identifiers
  - **Literals**
    - Plain or typed values
      - `"John"`
      - `"John"^^xsd:string`
    - Only allowed as objects

Statements

- Allowed structure of triples
  - **Subject**
    - *URI reference* or *blank node*
  - **Predicate**
    - *URI reference*
  - **Object**
    - *URI reference* or *blank node* or *literal*

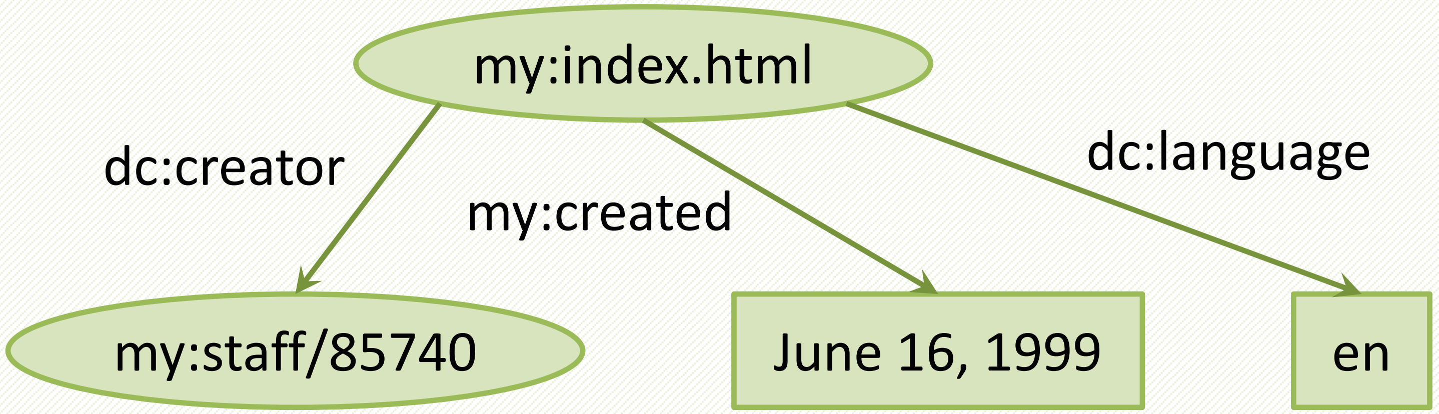
Data Model

- Data models
  - Relations
  - Trees
  - …
- RDF data model
  - **Directed labeled multigraph**
    - Vertices for subjects and objects
    - Labeled edges for particular triples

Data Model

- Example

```
my:index.html dc:creator my:staff/85740 .
my:index.html my:created "June 16, 1999" .
my:index.html dc:language "en" .
```



Serialization

- Existing approaches
  - **Triples notation**
    - Plain list of triples separated by dots
  - **RDF/XML**
    - Serialization to XML documents
  - **Turtle notation**
  - **Notation 3**

Triples Notation

- Syntax
  - Statements are terminated by dots
  - Triple parts are separated by spaces
  - URI references in angle brackets < >
  - Literal values in quotation marks " "
- Example

  -
    ```
    <http://is.cuni.cz/studium/sis#student358>
    <http://is.cuni.cz/studium/sis#name>
    "John" .
    ```

Blank Nodes

- **Structured properties**
  - Problem motivated by real-world data
  - Flat address…
    - `my:staff/8574 my:hasAddress "Malostranske nam. 25, 11800 Prague" .`
  - … vs. structured address

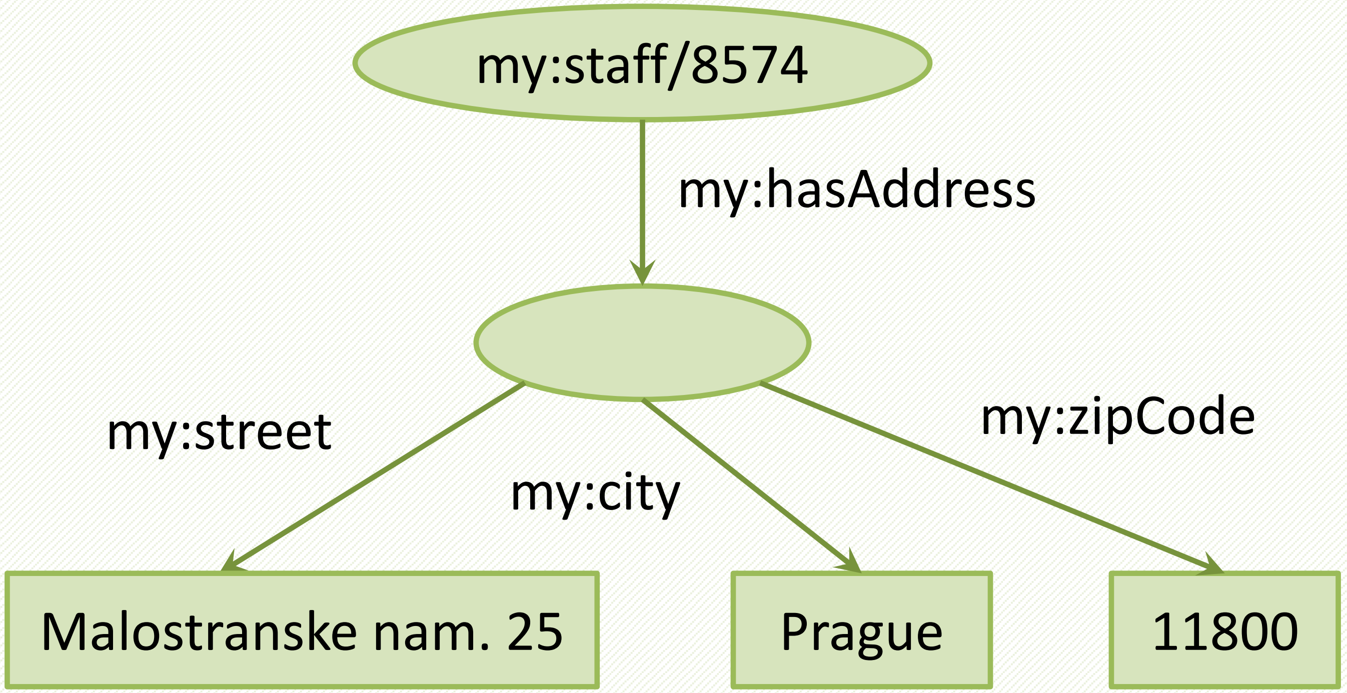
    -
      ```
      my:staff/8574 my:hasAddress my:address/1856 .
      my:address/1856 my:street "Malostranske nam. 25" .
      my:address/1856 my:city "Prague" .
      my:address/1856 my:zipCode "11800" .
      ```

Blank Nodes

- **Blank node identifiers**
  - Motivation
    - We use blank node identifiers instead of assigning standard and globally unique URI references
    - But we still need to ensure local uniqueness in a graph
  - Notation
    - `_:LocalName`
  - Graph drawings
    - **Blank nodes**, i.e. nodes without URI references

Blank Nodes

- Sample graph drawing



Typed Literals

- **Plain literals**
- **Typed literals**
  - Literal values are assigned with data types
  - These types are identified by URI references
  - RDF has no built-in set of data types of its own
    - XML Schema data types are commonly used

Typed Literals

- Examples
  - Plain literal
    - `my:index.html my:created "1999-06-16" .`
  - Typed literal
    - `my:index.html my:created "1999-06-16"^^ <http://www.w3.org/2001/XMLSchema#date> .`
    - `my:index.html my:created "1999-06-16"^^xsd:date .`

Typed Literals

- **XSD data types**
  - string
  - boolean
  - decimal, float, double
  - integer, positiveInteger, …
  - date, time, dateTime, gYear, …
  - …

Assignment 2.1

- Create an RDF graph for a simple student information system…
  - Describe a few particular students, courses, teachers, their relations and attributes
  - Use the following constructs:
    - Plain and typed literals
    - Blank nodes

Assignment 2.2

- Serialize the RDF graph from Assignment 2.1 using Triples notation…

Vocabularies

- **Vocabularies**
  - We already know how statements are used to describe things identified by URI references…
  - … but we also need a way to describe terms we intend to use in these statements
  - In particular:
    - Types of things (rdf:type)
    - Properties, their domains and ranges…
    - …

Vocabularies

- Well-known vocabularies
  - **RDF**
    - rdf: = http://www.w3.org/1999/02/22-rdf-syntax-ns#
  - **RDFS**
    - rdfs: = http://www.w3.org/2000/01/rdf-schema#
  - **OWL**
    - owl: = http://www.w3.org/2002/07/owl#
  - **Dublin Core**
    - dc: = http://purl.org/dc/elements/1.1/
  - …

RDF/XML

- **RDF/XML**
  - Normative syntax for writing RDF
  - W3C recommendation
    - http://www.w3.org/TR/rdf-syntax-grammar/
- Output pattern

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   …>
   …
</rdf:RDF>
```

RDF/XML

- Basic syntax

```
<rdf:RDF …>
  <rdf:Description rdf:about="SubjectReference">
    <PredicateName>ObjectLiteral</PredicateName>
    <PredicateName rdf:resource="ObjectReference"/>
    …
  </rdf:Description>
  …
</rdf:RDF>
```

RDF/XML

- Blank nodes

```
<rdf:RDF …>
  <rdf:Description rdf:about="SubjectReference">
    <PredicateName rdf:nodeID="BlankNodeIdentifier"/>
  </rdf:Description>
  <rdf:Description rdf:nodeID="BlankNodeIdentifier">
    …
  </rdf:Description>
  …
</rdf:RDF>
```

RDF/XML

- Typed literals

```
<rdf:Description rdf:about="SubjectReference">
  <PredicateName rdf:datatype="LiteralTypeReference">
    ObjectLiteral
  </PredicateName>
</rdf:Description>
```

- Alternative way

```
<!DOCTYPE rdf:RDF [<!ENTITY t "TypeRefPrefix">]>
…
<… rdf:datatype="&t;TypeLocalName">…</…>
```

Assignment 2.3

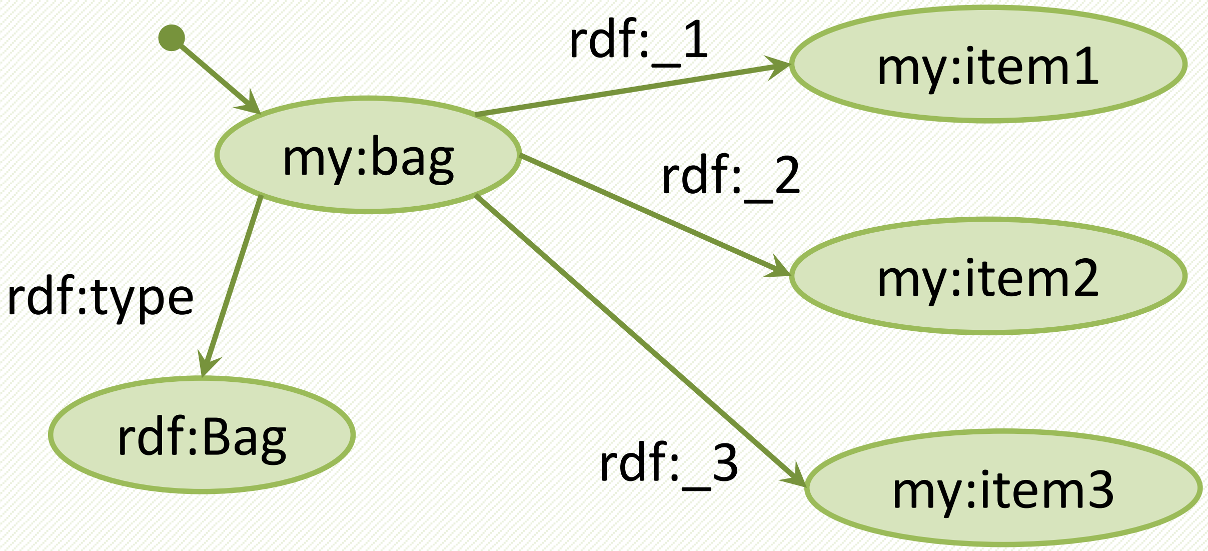
- Serialize the RDF graph from Assignment 2.1 using RDF/XML notation…

Containers

- **Containers**
  - Represent groups of resources or literals
- Available container types
  - **Bag** (rdf:Bag)
    - Without ordering and possibly with duplicates
  - **Sequence** (rdf:Seq)
    - With ordering and possibly with duplicates
  - **Alternative** (rdf:Alt)
    - Group of things that are alternative to each other

Containers

- Sample bag container



Containers

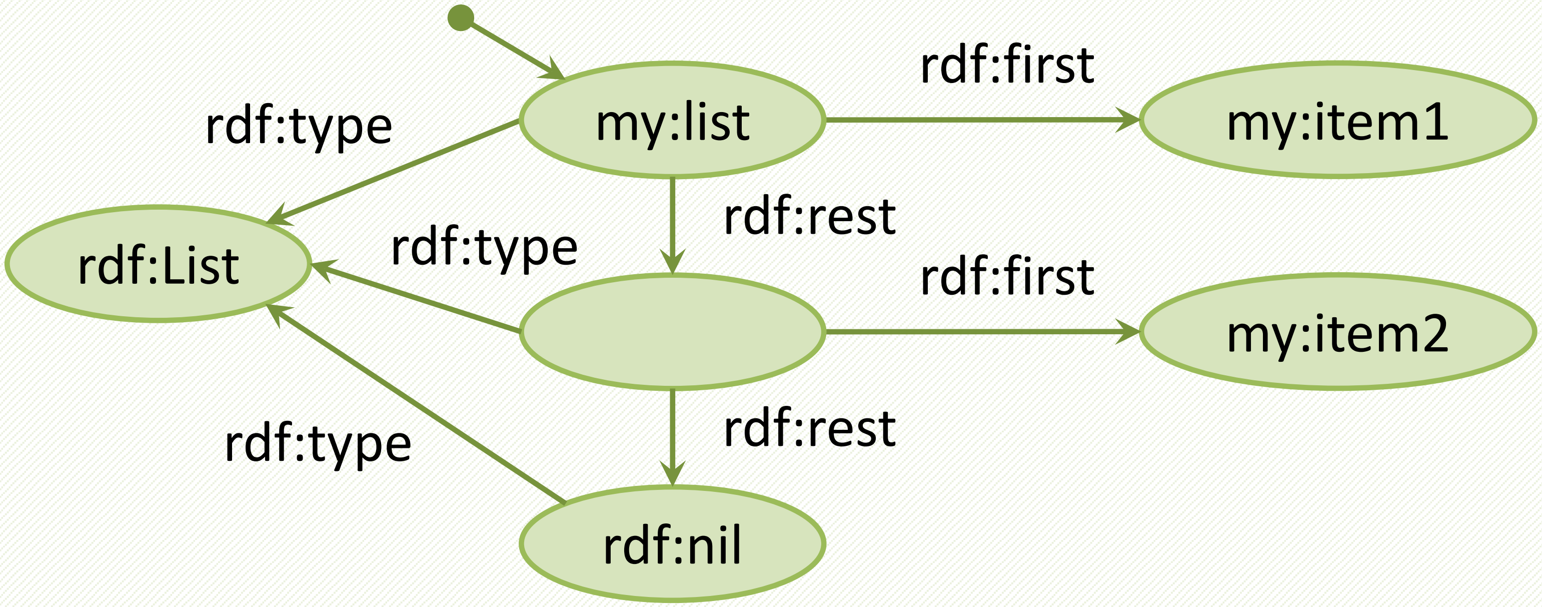
- Semantics
  - Note that any special meanings associated with containers are only intended meanings!
    - E.g. that items of *Alt* are really alternatives
  - Next, there is no way to close containers
    - I.e. to say that there are no other members

Collections

- **Collections**
  - rdf:List
  - Allow to close intended group members
  - Recursive construction
    - rdf:first
    - rdf:rest

Collections

- Sample list collection



Reification

- **Statements about statements**

```
my:s        my:p           my:o           .

_:triple1   rdf:type       rdf:Statement  .
_:triple1   rdf:subject    my:s           .
_:triple1   rdf:predicate  my:p           .
_:triple1   rdf:object     my:o           .
```

Turtle Notation

- **Terse RDF Triple Language**
  - Subset of Notation 3
  - http://www.w3.org/TeamSubmission/turtle/
- Example

```
@prefix my: <http://www.my.org/> .
my:s1 my:p1 my:o1 ,
            my:o2 ;
      my:p2 [ my:p3 my:o3 ] .
```

Assignment 2.4

- Serialize the RDF graph from Assignment 2.1 using Turtle notation…

Conclusion

- **RDF notions**
  - Resource, URI reference, blank node, literal
  - Statement: subject, predicate and object
  - Plain and typed literals
  - Containers and collections
  - Reification
- **Serialization**
  - Triples notation, RDF/XML, Turtle