Turtle – RDF Graph

Damir Cavar

January 2025

Turtle – Basic Syntax

- Triples are terminated with a full stop
- URLs are encoded in angle brackets (< and >)
- Literals are enclosed by double quotes
- <http://example.com/thing> <http://example.com/relation> "Some Text".

Turtle - Prefixes

- Use @PREFIX to shorten URLs
  - @PREFIX ex: <http://example.com>
  - Which enables us to write
  - ex:thing ex:relation “some text”

Turtle – Triples about same subject

```
@PREFIX ex: <http://example.com/> .
ex:thing ex:relation "Some Text" .
ex:thing ex:otherrelation ex:otherthing .
```

- can be written as:

```
@PREFIX ex: <http://example.com/> .
ex:thing ex:relation "Some Text" ;
           ex:otherrelation ex:otherthing .
```

Turtle – same properties

```
@PREFIX ex: <http://example.com/> .
ex:thing ex:relation "Some Text" .
ex:thing ex:relation ex:otherthing .
```

- can be written as:

```
@PREFIX ex: <http://example.com/> .
ex:thing ex:relation "Some Text" , ex:otherthing .
```

Turtle – Eliminate Redundant Triples

```
@PREFIX ex: <http://example.com/> .
ex:thing ex:relation "Some Text" .
ex:thing ex:relation "Some Text" .
```

has same meaning as:

```
@PREFIX ex: <http://example.com/> .
ex:thing ex:relation "Some Text" .
```

Turtle – blank nodes

```
@PREFIX ex: <http://example.com/> .
 _:a ex:relation "Some Text" .
```

- 'a' is the label - valid only within a single document
- if above triple appeared in another document it would refer to different node

Turtle – unlabelled blank nodes

```
@PREFIX ex: <http://example.com/> .
  ex:thing ex:relation _:a .
  _:a ex:property "foo" .
  _:a ex:property "bar“ .
```

is same as

```
ex:thing ex:relation [
  ex:property "foo" ;
  ex:property "bar" ] .
```

Turtle – literals with language

- In RDF, literals can have a language
- Written in Turtle as:

@PREFIX ex: <http://example.com/> . ex:thing ex:relation "Hello"@en .

ex:thing ex:relation "Bonjour"@fr .

Turtle – literal with datatypes

- In RDF, literals can have a datatype
- Written in Turtle as:

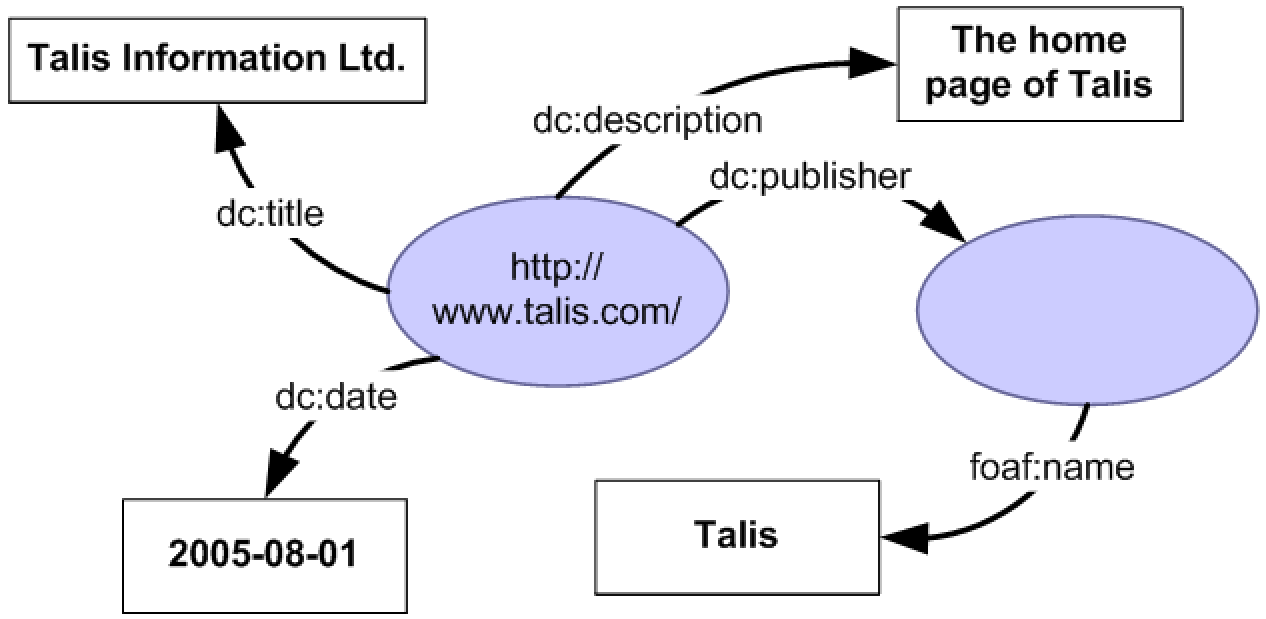
  @PREFIX ex: <http://example.com/> . ex:thing ex:relation "49"^^<http://example.com/datatype> .
- Can't have both a datatype and a language

Turtle – Longer example

```
@PREFIX dc: <http://purl.org/dc/elements/1.1/> .
@PREFIX foaf: <http://xmlns.com/foaf/0.1/ . <http://www.talis.com/>
   dc:title "Talis Information Ltd." ;
   dc:description "The home page of Talis" ;
   dc:publisher [
       foaf:name "Talis"
   ] ;
  dc:date "2005-08-01" .
```

- Interpreted as... the resource denoted by the URI http://www.talis.com/ has a title ..., a description ..., was published by ...

Turtle – Longer example



Turtle - Types

- 'a' keyword is shorthand for the URI http://www.w3.org/1999/02/22-rdf-syntax-ns#type

```
@PREFIX dct: <http://purl.org/dc/terms/> .
_:x
      a dct:Collection .
```

- Same as

```
@PREFIX dct: <http://purl.org/dc/terms/> . @PREFIX rdf: .
_:x
      rdf:type dct:Collection .
```

More on Turtle schemas

- By convention properties are named using camel case: theProperty
- Classes are named using title case: TheClass
- Not universal, just a convention

Turtle Schema example

- Suppose we have this RDF schema:

```
@PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@PREFIX ex: <http://example.com/schema#> .

 ex:Person a rdfs:Class .
 ex:spouse a rdfs:Property .
```

Turtle Schema example

- We could use it like this:

```
@PREFIX ex: <http://example.com/schema#> .
_:fred
  a ex:Person ;
  ex:spouse _:wilma .
```

- A query for all things with type ex:Person would return fred

Adding a range

```
ex:Person a rdfs:Class .
ex:spouse a rdfs:Property ;
          rdfs:range ex:Person .
```

- Now whenever we use the property ex:spouse we can infer that the value is a ex:Person
- A query against the data will now return wilma as well.

Adding a domain

- We can simplify by adding a domain for the property

```
ex:Person a rdfs:Class .
ex:spouse a rdfs:Property ;
          rdfs:range ex:Person .
          rdfs:domain ex:Person .
```

- Which lets us omit the type from our data - we can infer it instead

```
@PREFIX ex: <http://example.com/schema#> . _:fred
    ex:spouse _:wilma .
```