

Music and the Web of Linked Data

This is the quick reference guide associated with the ISMIR 2009 tutorial PM2. It provides some basic information on RDF/Turtle as well as the SPARQL query syntax, a listing of music-related data sets, and a glossary of terms.

Additional information can be found on the tutorial website: <http://ismir2009.dbtune.org>

RDF/Turtle

Turtle or Terse RDF Triple Language is an RDF serialization format that can be easily parsed by both humans and machines.

Namespaces:

To declare a new prefix, use the following notation:

```
@prefix prefix: <full http namespace> .
```

For example:

```
@prefix mo: <http://purl.org/ontology/mo/> .
```

Whenever you use `mo:something` in your RDF document, the parser will interpret it as `http://purl.org/ontology/mo/something`.

A set of namespaces used throughout the tutorial is given at the end of this document.

RDF Triples:

The simplest way of writing an RDF triple in Turtle is the following:

```
<URI of subject> <URI of property> <URI of object> .
```

Using namespaces defined with the `@prefix` notation, it results in *compact* URIs:

```
:artist rdf:type mo:MusicArtist .
```

You can also use a **literal** as an object:

```
:artist foaf:name "Artist's name" .
```

or specify the language:

```
:artist foaf:name "Artist's name"@en .
```

Note the above two literals are *not* equivalent because of the language tag.

You can explicitly **type** literals as well:

```
:artist foaf:name "Artist's name"^^xsd:string .
:artist foaf:birthday "1980-06-12"^^xsd:date .
```

Blocks of triples:

In order to write several triples at one, you can write them in **blocks**.

Here, we write two statements about the same subject. `:work` is a musical work, and it has a title.

```
:work
  a mo:MusicalWork ;
  dc:title "Franz Schubert's Trout Quintet" .
```

Here, we write multiple objects for a single subject/property pair. Mozart's requiem has multiple movements.

```
:requiem mo:movement :introitus, :kyrie, :sequenz, :offertorium,
           :sanctus, :benedictus, :agnus-dei, :communion.
```

Anonymous (or "blank") nodes:

In RDF, an element of a triple (subject, property, or object) does not have to have a URI. In that case, we call the element a **blank node**. Blank nodes in Turtle can be written using square brackets.

```
:yves foaf:knows [ foaf:name "Kurt" ] .
```

Here, we expressed "Yves knows someone named Kurt" or "There exist someone named Kurt that Yves knows"

Typing:

In RDF, we tend to use the property `rdf:type` a lot, to declare the type of a particular resource. In Turtle, the keyword `a` can be used instead of `rdf:type`.

```
:artist a mo:MusicArtist .
```

is the same statement as:

```
:artist rdf:type mo:MusicArtist .
```

Validation:

An online RDF/Turtle validator is available at <http://www.rdfabout.com/demo/validator/>

The command line tool "rapper" (part of the raptor-utils package in Debian and Ubuntu) parses all RDF syntaxes.

Example:

Here is a full RDF/Turtle example, where we describe a performance and the corresponding recording of the Franz Schubert's Trout Quintet.

```
@prefix mo: <http://purl.org/ontology/mo/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <#> .

:work
  a mo:MusicalWork ;
  dc:title "Franz Schubert's Trout Quintet" .

:performance
  a mo:Performance ;
  dc:title "Trout Quintet, performed by the London Symphony Orchestra" ;
  mo:performance_of :work ;
  mo:performer <http://dbpedia.org/resource/London_Symphony_Orchestra> ;
  mo:recorded_as [
    a mo:Signal ;
    mo:published_as :track1 ;
    dc:title "Recording of the LSO performing the Trout Quintet" ;
  ] .

:track1
  a mo:Track ;
  mo:track_number 5 ;
  owl:sameAs <http://dbtune.org/musicbrainz/resource/track/3208fbce-c20f-4362-
a3d5-5405ac1904bd> ;
  dc:title "Trout Quintet, performed by the LSO, on 'Favorite Classics'" .

# note that white space, indents and newlines are generally for readability
# also note a single-line comment begins with a hash like this one
```

SPARQL

SPARQL (pronounced "sparkle") is a W3C Recommendation that allows one to perform complex joins of disparate databases in a single, simple query and query for triple patterns using conjunction, disjunction, and optional pattern. It is the preferred method for querying a collection of RDF data.

Basic syntax:

Prologue (optional)	BASE <uri> PREFIX <i>prefix</i> : <uri> (repeatable)
Query result forms (required, pick 1)	SELECT ? <i>variable1</i> ? <i>variable2</i> ? <i>variable3</i> SELECT DISTINCT ? <i>variable1</i> ? <i>variable2</i> SELECT DISTINCT * DESCRIBE ? <i>variable</i> DESCRIBE <uri> DESCRIBE * CONSTRUCT { <i>graph pattern</i> } ASK
Query dataset sources (optional but recommended)	add triples to the query dataset (repeatable): FROM <uri> add a named graph to the query dataset (repeatable): FROM NAMED <uri>
Graph pattern (technically optional, required for ASK, but one almost always included)	WHERE { <i>graph pattern</i> } WHERE { <i>graph pattern</i> FILTER (<i>expression</i>) } WHERE { <i>graph pattern</i> OPTIONAL { <i>graph pattern</i> } } WHERE { { <i>graph pattern</i> } UNION { <i>graph pattern</i> } } WHERE { GRAPH < <i>graph_uri</i> > { <i>graph pattern</i> } } WHERE { GRAPH ? <i>variable</i> { <i>graph pattern</i> } }
Query result ordering (optional)	ORDER BY ASC(? <i>variable</i>) ORDER BY DESC(? <i>variable</i>)
Query results selection (optional)	LIMIT <i>n</i> OFFSET <i>n</i>

More on graph patterns:

Graph patterns use a Turtle-like syntax. The most notable addition is that of the variable denoted by `$variable` or more commonly `?variable`. In the following we perform a conjunction where all patterns must hold to return a match (note the `rdf` and `foaf` prefixes are assumed):

```
{ ?person rdf:type ?type .  
  ?person foaf:name ?name . }
```

To return a match `?person` must have both an `rdf:type` and a `foaf:name` specified. We can make the `foaf:name` condition optional as follows:

```
{ ?person a ?type .  
  OPTIONAL { ?person foaf:name ?name } }
```

Note that in the above `a` is a short cut for `rdf:type`. We can also create a disjunction (or) pattern using the `UNION` keyword.

```
{ {?person a mo:MusicArtist} UNION
  {?person a dbpo:MusicalArtist} }
```

The above will return a match for `?person` when the type is `mo:MusicArtist` or `dbpo:MusicalArtist`

Values - datatypes, expressions and operators:

The `FILTER` keyword maybe included with a graph pattern to filter results by value. A variety of operators maybe applied including

- logical operators: `A||B`, `A && B`, `!A`, `(A)`
- comparison operators: `=`, `!=`, `<`, `>`, `<=`, `>=`
- unary and binary operators: `+A`, `-A`, `A+B`, `A-B`, `A*B`, `A/B`
- RDF boolean operators: `BOUND(A)`, `isURI(A)`, `isBLANK(A)`, `isLITERAL(A)`
- RDF casting operators: `STR(A)`, `LANG(A)`, `DATATYPE(A)`

The `REGEX` string matching operator may also be used in a filter:

```
REGEX ( string expression , pattern expression [, flags ] )
```

Where *pattern* follows the Perl-like XQuery 1.0 schema. Flags include `s`, `m`, `i`, `x` . Supported datatypes include: RDF terms, `xsd:boolean`, `xsd:string`, `xsd:double`, `xsd:float`, `xsd:decimal`, `xsd:integer` and `xsd:dateTime`

Example SPARQL queries:

This query can be performed against the DBpedia endpoint at <http://dbpedia.org/sparql> and states "Find all music artists who do not play guitar and, if they're dead, give their death date; order results by instrument"

```
# first we define a prefix
PREFIX dbpo: <http://dbpedia.org/ontology/>

SELECT DISTINCT ?artist ?instrument ?died
FROM <http://dbpedia.org>
WHERE {
  ?artist a dbpo:MusicalArtist ;
  dbpo:instrument ?instrument .
  OPTIONAL { ?artist dbpo:deathdate ?died } .
  # the "i" flag below specifies case insensitive
  FILTER ( !REGEX ( STR(?instrument), "guitar", "i" ) )
}
ORDER BY ASC (?instrument) LIMIT 10 OFFSET 20
```

This query asks an endpoint to list the properties and types that are contained in the store.

```
SELECT DISTINCT ?prop ?type
WHERE {
  { [] ?prop [] . } UNION
  { [] a ?type . }
}
```

Available datasets

A listing of music-related datasets and how they can be accessed.

Musicbrainz endpoint

Description: A large and well-linked sparql endpoint for musicbrainz.org covering data about music artists and their releases

URIs: <http://dbtune.org/musicbrainz/> - human readable splash page
<http://dbtune.org/musicbrainz/sparql> - SPARQL URI
<http://dbtune.org/musicbrainz/snorql/> - AJAX Snorql SPARQL query explorer

BBC John Peel sessions

Description: A sparql endpoint for the BBC John Peel sessions, holding data about each session, the artists playing and the instruments they played, the tracks played, the sound engineers, the studio used, etc.

URIs: <http://dbtune.org/bbc/peel/> - human readable splash page
<http://dbtune.org/bbc/peel/sparql/> - SPARQL URI

Magnatune

Description: A sparql endpoint for magnatune.com covering data about music artists and their releases on the Magnatune label

URIs: <http://discogs.dataincubator.org/.html> - human readable splash page
<http://dbtune.org/magnatune/sparql/> - SPARQL URI

Jamendo endpoint

Description: A sparql endpoint for jamendo.com covering data about music artists and releases in this Creative Commons label

URIs: <http://dbtune.org/jamendo/> - human readable splash page
<http://dbtune.org/jamendo/sparql/> - SPARQL URI

Discogs endpoint

Description: A well-linked sparql endpoint for discogs.com covering data about music artists and their releases

URIs: <http://discogs.dataincubator.org/.html> - human readable splash page
<http://api.talis.com/stores/discogs/services/sparql> - SPARQL URI

BBC endpoints

Description: A sparql endpoint with data from BBC websites

URIs: <http://api.talis.com/stores/bbc-backstage/services/sparql> - Talis-backed endpoint
<http://bbc.openlinksw.com/sparql> - Openlink Virtuoso-backed endpoint

Myspace endpoint

Description: A sparql endpoint containing some data about Myspace profiles

URIs: <http://dbtune.org/myspace/> - human readable splash page
<http://virtuoso.dbtune.org/sparql> - SPARQL URI and interface

DBpedia.org endpoint

Description: A sparql endpoint for dbpedia.org covering the info box content in wikipedia

URIs: <http://dbpedia.org/sparql> - SPARQL URI and interface
<http://dbpedia.org/snorql> - AJAX Snorql query interface

Additional datasets become available all the time. For updates to this list see:
<http://ismir2009.dbtune.org/taxonomy/term/5>

Namespaces used in the tutorial

The following prefixes and namespaces are used in the tutorial:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix geo: <http://www.geonames.org/ontology#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix p: <http://dbpedia.org/property/>.
@prefix dbpo: <http://dbpedia.org/ontology/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix dc: <http://purl.org/dc/elements/1.1/>.
@prefix mo: <http://purl.org/ontology/mo/>.
@prefix event: <http://purl.org/NET/c4dm/event.owl#>.
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#>.
@prefix to: <http://purl.org/ontology/tonality/>.
@prefix po: <http://purl.org/ontology/po/>.
@prefix so: <http://purl.org/ontology/symbolic-music/>.
@prefix vamp: <http://purl.org/ontology/vamp/>.
@prefix list: <http://www.w3.org/2000/10/swap/list#>.
@prefix wgs: <http://www.w3.org/2003/01/geo/wgs84_pos#>.
@prefix yago: <http://dbpedia.org/class/yago/>.
@prefix tags: <http://www.holygoat.co.uk/owl/redwood/0.1/tags/>.
@prefix : <#>.
```

Glossary

A brief glossary of terms.

Document (or information resource) - A Web resource that can be transmitted electronically e.g. your Web page as opposed to a non-information resource which cannot be sent over the wire.

Linked data - HTTP + URI + RDF

Linking Open Data (LOD) - A community project running since late 2006, publishing open data as linked data

N3 - An extension to the RDF data model with an associated serialisation, from which RDF/Turtle is derived

Namespace - A Web location under which several resources are defined

Ontology (or vocabulary) - A namespace defining concepts and relationships in a particular domain

OWL (Web Ontology Language) - A set of constructs for defining web ontologies

RDF - a simple data model for publishing data on the Web

RDF/N-Triples - A very simple text RDF serialisation (writing one triple after the other)

RDF/Turtle - N-Triples with some syntactic sugar, a subset of N3, N3 minus the extensions to RDF, our favorite RDF serialization

RDF/XML - An XML serialisation for the RDF data model, not very human readable or concise but very well supported

SPARQL - A query language and protocol for RDF data

Statement (or triple) - A (subject, property, object) tuple, e.g. ('this tutorial', 'is presented by', 'Kurt')

Thing (or non-information resource) - A resource that can be named by a URI but which cannot be transmitted electronically. Human beings, abstract concepts, etc. are non-information resources

URI - Uniform Resource Identifier, a standard for identifying and locating resources on the internet