Sandboxing your data with Tracker

Carlos Garnacho
RDF

- Resource Description Framework
RDF

- Resource Description Framework
- Describes graphs of information
RDF

• Resource Description Framework
• Describes graphs of information
• A concept, not a format
  • Not XML
  • Not TTL
  • Not JSON-LD
  • ...
  • Many ways to represent RDF
RDF

- Resource Description Framework
- Describes graphs of information
- A concept, not a format
  - Not XML
  - Not TTL
  - Not JSON-LD
  - …
  - Many ways to represent RDF
- Its basic unit is a triple
Triples?
Triples

- Subject
Triples

- Subject
- Predicate
Triples

- Subject
- Predicate
- Object
Triples

<subject> <predicate> <object> .
Triples

\(<\text{subject}>\quad <\text{predicate}>\quad <\text{object}>\quad .\)
Triples

\[ <\text{subject}> \ <\text{predicate}> \ <\text{object}> . \]
Triples

\(<\text{subject}\> \ <\text{predicate}\> \ <\text{object}\> \ .\)
Triples

<subject> <predicate> <object> .

URI
Blank node

URI
Triples

\(<\text{subject}\> \ <\text{predicate}\> \ <\text{object}\> .\)
Triples

\(<\text{subject}>\)  \(<\text{predicate}>\)  \(<\text{object}>\).
Triples

subject → predicate → object
Triples work by composition
<song1> a ex:Song .
<song1> a ex:Song .
<song1> ex:duration 285 .
<song1> a ex:Song .
<song1> ex:duration 285 .
<song1> ex:title ‘Why’ .

song1

条评论

ex:duration

285

ex:title

Why
<song1> a ex:Song .
<song1> ex:duration 285 .
<song1> ex:title 'Why' .
<song1> ex:artist <artist1> .
<song1> a ex:Song .
<song1> ex:duration 285 .
<song1> ex:title 'Why' .
<song1> ex:artist <artist1> .
<artist1> a ex:artist .
<song1> a ex:Song .
<song1> ex:duration 285 .
<song1> ex:title ‘Why’ .
<song1> ex:artist <artist1> .
<artist1> a ex:artist .
<artist1> ex:name ‘Joe’ .
\(<\text{song1}\> \text{ a ex:Song .}\\ \<\text{song1}\> \text{ ex:duration 285 .}\\ \<\text{song1}\> \text{ ex:title ‘Why’ .}\\ \<\text{song1}\> \text{ ex:artist } \<\text{artist1}\> .\\ \<\text{artist1}\> \text{ a ex:artist .}\\ \<\text{artist1}\> \text{ ex:name ‘Joe’ .}\\ \<\text{song1}\> \text{ ex:inFile } \<\text{file1}\> .\)
<song1> a ex:Song .
<song1> ex:duration 285 .
<song1> ex:title 'Why' .
<song1> ex:artist <artist1> .
<artist1> a ex:artist .
<artist1> ex:name 'Joe' .
<song1> ex:inFile <file1> .
<file1> ex:containerOf <song1>.
But the structure has constraints
Ontologies
Ontologies

• Definition of the allowed data
Ontologies

• Definition of the allowed data
  • Classes
Ontologies

- Definition of the allowed data
  - Classes
  - Properties
Ontologies - range

Song \(\rightarrow\) ex:duration \(\rightarrow\) 285
Ontologies - cardinality

Song \(\xrightarrow{\text{ex:duration}}\) 285
Ontologies - cardinality
Ontologies

• Also:
  • Indexes
Ontologies

• Also:
  • Indexes.
  • Documentation
Ontologies

• Also:
  • Indexes.
  • Documentation
  • ...

...
Ontologies

- Also:
  - Indexes.
  - Documentation
  - ...

- Ontologies are defined on top of RDF
  - Introspectable
SPARQL

- Query language for RDF graphs
SPARQL

- Query language for RDF graphs
- W3C recommendation from 2013
SPARQL

- Query language for RDF graphs
- W3C recommendation from 2013
- Works by specifying „Graph Patterns“
SELECT ?u {
  ?u ex:title 'Why'
}
SELECT ?t {
    <song1> ex:title ?t
}

file1

song1

285

Why

artist1

Joe

SELECT ?t {
    <song1> ex:title ?t
}
SELECT ?u ?t {
  ?u ex:title ?t
}

file1

song1

285

Why

artist1

Joe

ex:inFile

rdf:type

ex:duration

ex:title

ex:artist

ex:name
SELECT ?u ?p {
    ?u ?p 'Why'
}
SELECT ?u {
  ?a ex:name 'Joe'
}
SELECT ?u {
  ?u ex:artist [ ex:name 'Joe' ]
}
SELECT ?u {
  ?u ex:artist/
  ex:name 'Joe'
}
SPARQL

- Query language for RDF graphs
- W3C recommendation from 2013
- Works by specifying „Graph Patterns“
- Can work on multiple graphs
  - Anonymous graph
  - Named graphs
  - Unions of them
SPARQL

- Query language for RDF graphs
- W3C recommendation from 2013
- Works by specifying „Graph Patterns“
- Can work on multiple graphs
  - Anonymous graph
  - Named graphs
  - Unions of them
- Allows federated queries, can work across services
SPARQL is a language made for an unreliable web.
Tracker
Tracker

- Implementation of a triple store w/ SPARQL 1.1
Tracker

- Implementation of a triple store w/ SPARQL 1.1
- Compact, fast, versatile
Tracker

- Implementation of a triple store w/ SPARQL 1.1
- Compact, fast, versatile
- Suitable for private data
Endpoints and portals
Endpoints and portals

- Network D-Bus
Endpoints and portals

- Network D-Bus
- Unreliable Constrained
Endpoints and portals

- Network D-Bus
- Unreliable Constrained
  - Services, and graphs in them
Endpoints and portals

- **Network D-Bus**
- **Unreliable Constrained**
  - Services, and graphs in them
  - Defined in ".flatpak-info"
And more...

- Data dumps, serialization...
  - Import/export
  - Data exchange
And more...

- Data dumps, serialization...
  - Import/export
  - Data exchange
  - ...
- Network transparency
  - Data migrations
  - Automated synchronization
  - ...
Questions?

#tracker at irc.gnome.org
https://discourse.gnome.org