Supporting the Semantic Web and Linked Data

Overview

- Exposing
  - Dereferenceable http URIs for resources
  - Relationships using dereferenceable http URIs
- ... made possible by the new REST API
- Extending the scope of relationships in the resource index

Current situation - identifiers

- Identifiers used
  - **`namespace:pid`**
  - **`info:fedora/namespace:pid`**
  - **`http://server:port/fedora/get/namespace:pid`**
  - **`http://server:port/objects/namespace:pid`**
- Issues
  - identifier scope
  - canonical

Current situation - relationships

- Single graph
- Fedora objects (or datastreams) as subjects
- Identifiers used are info:fedora/
- No support for “arbitrary” RDF – eg “lifting” of XML metadata
- Specification of relationships are in imperative code

Resource Identifiers[1, 2]

- Deprecate the “LITE” APIs (/get)
  - HTTP 301: Moved permanently
  - then remove in future release
- Define canonical dereferenceable URIs
  - using the REST API URIs

Support http URIs in relationships

[6]

- Relationship:
- **<info:fedora/ns:pid1> <#isMemberOf> <info:fedora/ns:pid2>**
- Exposed as:
- **<http://server/fedora/objects/ns:pid1> <#isMemberOf>** http://server/fedora/objects/ns:pid2
- Query / results rewriting?
- Retain info:fedora for local/internal use
  - **/risearch?type=tuples&query=...&scope=local|global**

Support “arbitrary” RDF [3]

- Issue
  - create: myns:pid1 : **<s1> <p1> <o1>**
  - create: myns:pid2 : **<s1> <p1> <o1>**
  - RI contains: **<s1> <p1> <o1>**
  - delete: myns:pid1
  - **<s1> <p1> <o1>** deleted but myns:pid2 still asserts it
- Solution
  - Named graphs

Named Graphs[3]

- **<#some:pid1>** : graph containing triples asserted by object some:pid1
- **<#some:pid2>** : graph containing triples asserted by object some:pid2
- **<#some:pidn>** : graph containing triples asserted by object some:pidn
- **<#ri>** : defined as a view containing the above graphs
- Queries run over **<#ri>**

Named Graphs [3]

- If the same triple is asserted by two different objects, it appears in two graphs
- Query results contain one instance of the triple
- some:pid1 deleted: triple still present in graph created for some:pid2

Mulgara and graphs [3]

- Mulgara Models (graphs) can be
  - A model containing triples
  - Definition of a “view”: union (or intersection) of other graphs
- Other triple stores?

Issues [3]

- Performance: Querying <#ri> involves querying a large number of underlying graphs
  - test
- Graph names
- “Pollution” of resource index with arbitrary triples
  - Separate graphs for
    - <#ri> : “core” triples
    - <#riUser> : “user” triples
    - <#riFull> : <#ri> UNION <#riUser>
- Free text graph(s)
- Triple Store support – MPTStore?
  - disable “arbitrary” graphs if triple store does not support?
- Hierarchy of graphs to use

Graph Hierarchy [3]

<#ri> - a view containing:

- <#some:pid> - object graph for some:pid, a view containing:
  - <#some:pid/properties> - object properties triples
  - <#some:pid/datastreams> - a view containing:
    - <#some:pid/datastreams/rels-ext> - rels-ext triples
    - <#some:pid/datastreams/rels-int> - rels-int triples
      - <#some:pid/datastreams/dc> - DC triples
      - <#some:pid/datastreams/{rdf datastream}> - triples from rdf datastream
      - <#some:pid/datastreams/{dsid}/properties> - datastream properties
- <#some:otherpid> - object graph for some:otherpid, a view containing:
  - <#some:otherpid/properties> - etc
  - <#some:otherpid/datastreams> - etc

Only object graphs necessary to support main requirement

Specifying triples for objects [4]

- Currently generated by code
  - object and datastream properties, “default” content model
  - “conversion” of DC to triples
  - RELS-EXT, RELS-INT

Declarative specification of triples

[4]

- Content model specifies which datastreams to index
  - RDF datastreams
  - XSLT/GRDDL etc for XML (and other) datastreams
  - Object methods producing RDF
- XSLT for object and datastream properties

Mechanism [4]

- System object methods for generating core triples
- User content model object methods for generating user triples
- eg expose through REST API
  - **`/objects/some:pid/relationships`**
- Update triple store using these methods
  - Move out of core DOManager code, implement using decorator?

REST API [6]

- GET /objects/some:pid/relationships
  - /objects/some:pid/datastream/DC/relationships?
  - Content negotiation (Accept: application/rdf+xml)
  - URI parameter (?format=rdf)
- Other verbs
  - POST: set of triples to add
  - DELETE: set of triples to delete
  - PUT: modification, eg SPARQL Update
- Generic methods
  - update “core triples” (easy to identify source)
  - update arbitrary (specified) datastream
  - potential overlap between RELS-EXT and arbitrary datastream
- Operate directly on objects (not on triple store)
- SOAP API

Finally...

- Fedora generally sits behind an application
- Resource identifers exposed by the application may not be Fedora resource URIs
- **`/library/display?&resourceID=some%3Apid`**