RDF Isolation API

RDF Next Step Workshop 2010
James Leigh, David Wood
Zepheira LLC
RDF Isolation API

- Need an API to...
  - Manage RDF services
  - Manage multiple RDF store states
  - Managing queries
  - Describe relationships between services
RDF Isolation API

- CRUD operations for services
- CRUD operations for named queries
- Hypermedia links resources together
- Cacheable and serial evaluation
- Simple named query parameters
- Distributed revision control
Hypermedia

- All resources are represented using
  - application/sparql-query
- Explicit linking
  - No “well-known locations”
- Queries link using SERVICE
- Services and graphs link using WITH
- Service descriptions may link to named queries using rdf+xml
HTTP/1.1 200 Ok
Content-Type: application/sparql-query.

DESCRIBE ?book
FROM </graph>
WHERE {
  SERVICE </service> {
  }
}
Cacheable & Serial Eval

- Named queries can be evaluated using cacheable GET requests
- Named queries can also be evaluated serially using POST requests
- Ad-hoc queries can be serially evaluated using POST requests to a service or graph
HTTP/1.1 200 Ok
Content-Type: text/turtle
Cache-Control: max-age=30
Age: 0

    dc:creator "A.N.Other" .
Simple Query Parameters

- Query parameter are explicitly identified
- Each query parameter is exclusively
  - Absolute IRI or relative IRI
  - plain literal with an explicit language
  - typed literal with an explicit datatype
- Only string values of literal labels and relative IRIs need to be passed for evaluation
- The BINDINGS clause can be used to combine query parameters
Simple Query Parameters

POST /query HTTP/1.1
Accept: text/turtle
Content-Type: application/x-www-form-urlencoded

author=A.N.Other
Distributed Revisions

- Services maybe composed of others
- Services may store only a delta against other services
- Anybody can create their own virtual service from an accessible service
- Services can reject merging deltas to enforce data consistency
Distributed Revisions

PUT /branch HTTP/1.1
Content-Type: application/sparql-query

INSERT { ?s ?p ?o }
WHERE {
  SERVICE </service> {
    ?s ?p ?o
  }
}
SELECT *
WHERE {
    ?subj dc:title "A new book";
    dc:creator "A.N.Other".
}
SELECT *
WHERE {
    { ?subj dc:title "A new book" }
    UNION { SELECT ?subj {} } 
    BINDINGS ?subj { (<book3>) (<book4>) }
    EXISTS { ?subj dc:creator "A.N. Other" }
}
Distributed Revisions

PUT /union HTTP/1.1
Content-Type: application/sparql-query-fed

INSERT { ?s ?p ?o }
WHERE {
    SERVICE <http://example.com/service> { 
        ?s ?p ?o
        FILTER regex(str(?s), "^http://example.com/", "i")
    }
    SERVICE <http://example.org/service> { 
        ?s ?p ?o
        FILTER regex(str(?s), "^http://example.org/", "i")
    }
}
Thanks