Resource Index Search

Introduction

The Resource Index Search Service (RISearch) is a web service that exposes the contents of a repository's Resource Index guide for outside use. This document introduces the use of this service through a web browser interface, then describes how to access it programmatically.

User Interface

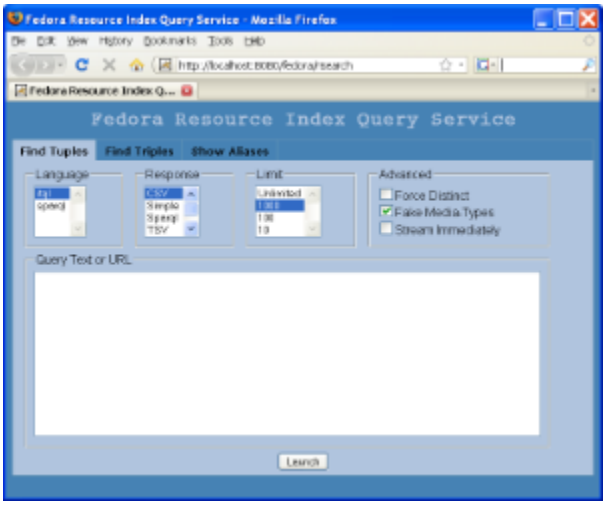
When your Fedora server is running, the RISearch service will be available under `/fedora/risearch`. For example:

```
http://localhost:8080/fedora/risearch
```

The user interface consists of three tabs: Find Tuples, Find Triples, and Show Aliases. A detailed description of each of these tabs follows.

Find Tuples

The "Find Tuples" tab shown below is used to run tuple queries against the resource index. A *tuple query* is one that returns a list of named values.



When you enter a query and click "Launch", a new browser window will display the results.

To get an idea of how it works, try the following iTQL query, which asks for information about all Service Definition objects in the repository:

```
select $object $modified from <#ri>
where  $object <fedora-model:hasModel> <info:fedora/fedora-system:ServiceDefinition-3.0>
and    $object <fedora-view:lastModifiedDate> $modified
```

In response, you should see something like this:

```
"object","modified"
info:fedora/demo:1,2009-02-16T19:39:28.859Z
info:fedora/demo:12,2009-02-16T19:39:17.843Z
info:fedora/demo:19,2009-02-16T19:39:20.375Z
info:fedora/demo:22,2009-02-16T19:39:20.671Z
info:fedora/demo:Collection,2009-02-16T19:39:24.89Z
info:fedora/demo:8,2009-02-16T19:39:34.281Z
info:fedora/demo:DualResolution,2009-02-16T19:39:25.093Z
info:fedora/demo:27,2009-02-16T19:39:26.578Z
```

This is a list of comma-separated values, each row representing the URI and modified date of the objects that matched the query.

Above the query text box, you can alter several settings for a query. These settings are described below.

Language

Indicates the query language to use. Currently, the options are **SPARQL** and **iTQL** (a full-featured RDF query language supported by Mulgara).

Response

Indicates the desired response format. Valid options include:

- **CSV** – Comma-separated values
- **Simple** – A simple easy-to-read text format that shows datatype information, when present
- **Sparql** – The W3C standard query response
- **TSV** – Tab-separated values
- **Count** – A count of the item returned by the query

Limit

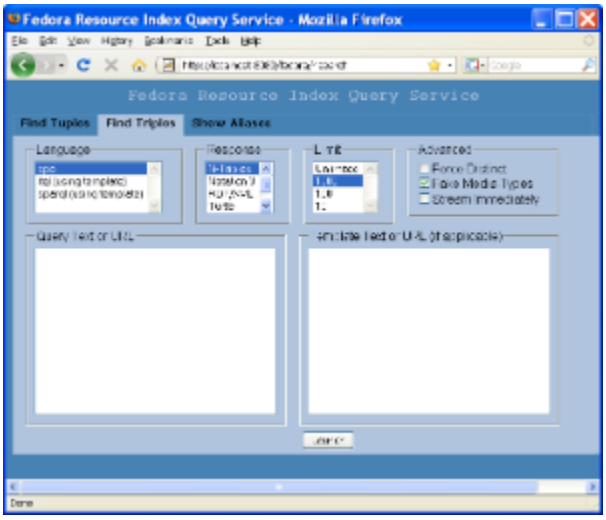
The maximum number of results to return. It is useful to set this low when testing queries.

Advanced

- **Force Distinct** – Whether to force duplicate results to be dropped. Note: iTQL never returns duplicates.
- **Fake Media-Types** – Whether to send incorrect Content-Type HTTP response headers with the responses (to trick browsers into displaying the results instead of popping up a "Save As/Open With" window).
- **Stream Immediately** – Whether to stream the results right away (faster), or to save them to a temporary file before sending them to the client. The default behavior (to save the results before streaming) will give a more informative error message if a query fails.

Find Triples

The "Find Triples" tab shown below is used to run triple queries against the resource index. A *triple query* is one that returns a list of RDF statements (aka triples).



This tab works in much the same way as the "Find Tuples" tab, but supports different response formats and provides a means to convert tuple query results to triples. It also exposes another query language: SPO.

SPO

This is a very simple RDF query language, where queries consist of a specific subject (or an asterisk, indicating "any"), a specific predicate (or an asterisk), and a specific object (or an asterisk). The easiest way to learn SPO is by example:

Get all triples in the repository

* * *

Get all triples where the object is demo:1

* * <info:fedora/demo:1>

Get all triples where the subject is demo:1 and the object is fedora-system:ServiceDefinition-3.0

<info:fedora/demo:1> * <info:fedora/fedora-system:ServiceDefinition-3.0>

Response Formats

A variety of RDF formats are supported:

- **N-Triples** – A subset of Notation 3 defined in the RDF Test Cases document
- **Notation 3** – The original RDF text format, defined by Tim Berners-Lee in An RDF language for the Semantic Web
- **RDF/XML** – The "RDF/XML" format, defined in the RDF/XML Syntax Specification
- **Turtle** – A newer subset of Notation 3, defined in Dave Beckett's Turtle - Terse RDF Triple Language
- **count** – A count of the item returned by the query

Using Templates

Templates are used to convert tuple query results to triples. A template consists of one or more *triple binding patterns* that reference the binding variables in an iTQL query.

The easiest way to understand how this works is by example.

In this example, we'll show how to extract a subgraph from the resource index using iTQL. Enter the following query text:

```
select $a $r $b from <#ri>
where  $a <fedora-model:hasModel> <info:fedora/fedora-system:FedoraObject-3.0>
and    $a $r $b
and    $b <fedora-model:hasModel> <info:fedora/fedora-system:FedoraObject-3.0>
```

This query by itself returns all relationships between data objects in a repository. The binding variables are $a, $r, and $b. Now enter the following in the template text box:

$a $r $b

When you launch the query, you'll see a list of triples: the sub-graph of all object-to-object relationships in the repository. (If you don't see anything, you should ingest the demo objects which include some sample relationships).

Now try the following template instead. This demonstrates how to derive new statements from those in the resource index:

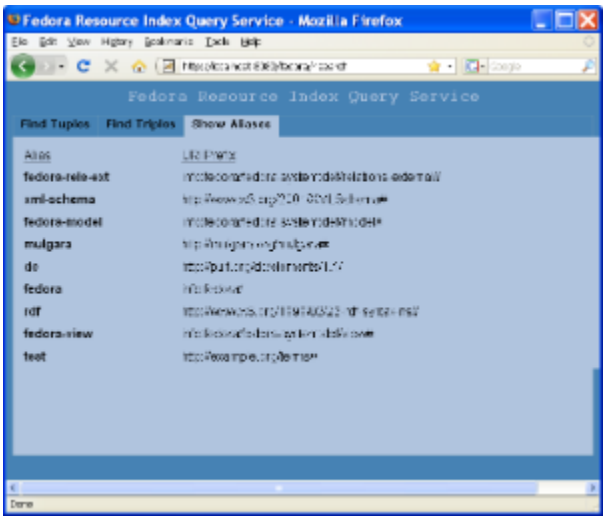
$a <urn:example:isRelatedTo> $b

$b <urn:example:isRelatedTo> $a

Running the query will now show *two* statements for every object-to-object relationship in the resource index graph.

Note: When using templates to transform tuples to triples, some duplicates may be returned. These can be avoided by checking "Force Distinct".

Show Aliases



This tab shows the aliases that can be used in queries and what URI prefixes they map to.

Aliases are just shortcuts that help make queries easier to write. For example, in a query you can write `<fedora-model:state>` instead of `<info:fedora/fedora-system:def/model#state>`.

Application Interface

The RISearch service can be programmatically accessed via HTTP GET or POST. To avoid character encoding issues, POST should always be used when the query is passed in by value and contains non-ASCII characters.

As with the user interface, it can be invoked to retrieve tuples or triples. The syntax is described below.

Note:

- Square brackets ("[" and "]") indicate that the parameter is optional.
- As with all HTTP parameters, unsafe URI characters should be URI-escaped. For readability purposes, URI escaping is not shown below.
- The *query* and *template* parameters optionally take the value by *reference* – that is, a URL to a query or template can be given instead of the actual text.
- The *flush* parameter tells the resource index to ensure that any recently-added/modified/deleted triples are flushed to the triplestore before executing the query. This option can be desirable in certain scenarios, but for performance reasons, should be used sparingly when a process is making many API-M calls to Fedora in a short period of time: We have found that Mulgara generally achieves a much better overall update rate with large batches of triples.

Syntax for Requesting Tuples

```
http://localhost:8080/fedora/risearch?type=tuples
                                     &flush=[*true* (default is false)]
                                     &lang=*itql|sparql*
                                     &format=*CSV|Simple|Sparql|TSV*
                                     &limit=[*1* (default is no limit)]
                                     &distinct=[*on* (default is off)]
                                     &stream=[*on* (default is off)]
                                     &query=*QUERY_TEXT_OR_URL*
```

Syntax for Requesting Triples

```
http://localhost:8080/fedora/risearch?type=triples
                                     &flush=[*true* (default is false)]
                                     &lang=*spo|itql|sparql*
                                     &format=*N-Triples|Notation 3|RDF/XML|Turtle*
                                     &limit=[*1* or more (default is no limit)]
                                     &distinct=[*on* (default is off)]
                                     &stream=[*on* (default is off)]
                                     &query=*QUERY_TEXT_OR_URL*
                                     &template=[*TEMPLATE_TEXT_OR_URL* (if applicable)]
```