Search indexing service

Purpose

Permits external applications to request specific updates to the VIVO search index, by providing a list of URIs whose search records may be out of date.

When the VIVO triple-store is updated in a way that bypasses VIVO's internal data channels, the search index will not reflect the updates.

With this service, you can provide a list of URIs whose contents have changed, and request that only those search records be updated. This is usually faster than rebuilding the entire index.

Use Cases

Use with ingest tools

The Harvester and similar tools write directly to the VIVO triple-store, bypassing the usual data channels in VIVO. After ingesting, it has been necessary to rebuild the search index so it will reflect the changes in the data. With this service, you can rebuild only part of the index.

Note: when the Harvester and other tools have been modified to use the SPARQL Update API, VIVO will ensure that the search index and inferences are kept in synchronization with the data.

Loading the triple-store

Some sites use two VIVO instances: a staging instance and a production instance. All ingests occur on the staging instance. Periodically, the triple-store is copied from staging to production. When this is done, you have 3 options:

- Copy the search index files from staging to production to keep it consistent with the triple-store
- Rebuild the search index in production
- Use the Search Indexing service to update specific records in the search index.

Indexing and Reasoning

The concerns that apply to the search index will also apply to the state of the inferred triples in the data model. When bypassing the data channels in VIVO, you bypass the semantic reasoner. To compensate for this, you must either

- Request that the reasoner rebuild all of the inferences, using `Recompute Inferences` from the `Site Administration` page, or
- Ensure that the ingested RDF contains all of the triples that you want VIVO to contain, including those that would be provided by the reasoner

In most cases, the time required to re-inference the model is greater than the time required to rebuild the search index. Unfortunately, the reasoning process is not easy to partition. To date, VIVO has no service that would allow you to update the inferences on a limited set of data.

Specification

URL

`[vivo]/searchService/updateUrisInSearch`

Examples:

```
http://vivo.cornell.edu/searchService/updateUrisInSearch
```

```
http://localhost:8080/vivo/searchService/updateUrisInSearch
```

HTTP Method

The API supports only HTTP POST requests with a content type of `multipart/form-data`.

If the request does not specify an encoding, UTF-8 is assumed.

Parameters

name	value
`email`	the email address of a VIVO administrator account
`password`	the password of the VIVO administrator account
`other`	One or more content parts, containing URIs to be indexed, separated by white space and/or commas

The name of the file content is unimportant. The API will examine all parts of the request and add any URIs to the list to be indexed. It is common, however, to put the entire list of URIs into a single content part.

Response Codes

Code	Reason
`200 OK`	Search indexing request was successful.
`403 Forbidden`	HTTP request did not include an `email` parameter.
	HTTP request did not include a `password` parameter.
	The combination of `email` and `password` is not valid.
	The selected VIVO account is not authorized to use the SPARQL Update API.
`500 Internal Server Error`	VIVO could not execute the request; internal code threw an exception.

Examples

This example uses the UNIX `curl` command to request updates to the search records of 3 individuals.

```
curl -v --form 'email=testAdmin@mydomain.edu' --form 'password=Password' --form 'uris=@uriList.txt'
'http://localhost:8080/vivo/searchService/updateUrisInSearch'
```

uriList.txt

```
http://vivo.mydomain.edu/individual/n6724
http://vivo.mydomain.edu/individual/n90987
http://vivo.mydomain.edu/individual/n32
```

Securing the API

The Search Indexing service is enabled by default. However, it is recommended that you secure the URL `/searchService` with HTTPS. Otherwise, email/password combinations will be sent across the network without encryption. Methods for securing the URL will depend on your site's configuration.