The navigation of logical file systems

Overview

The file system navigation of FOXpath is not restricted to the physical file system (the file system defined by your operation system). You can navigate also other types of "logical" file systems:

- archives (.zip and .jar)
- BaseX databases
- SVN repositories
- Github projects (via UTREE or UGRAPH, see below)
- UTREE file systems
- UGRAPH file systems

Navigation of archives (.zip, .jar)

You can navigate an archive by using URIs consisting of three parts:

- The location of the archive file
- A step consisting of the literal string #archive#
- Steps navigating into the archive

The #archive# step means "move to the roots of the archives located by the preceding step". This amounsts to moving from the file system containing the archive file into the file system represented by the archive file contents.

Examples

In the following examples, assume the existence of an archive file `/downloads/userguide.zip`.

Expression:

```
/downloads/userguide.zip/#archive#/*
```

Value: The top-level folders and files contained by the archive file.

Expression:

```
/downloads/userguide.zip/#archive#//file-ext() => frequencies(15)
```

Value: Frequency table of all file extensions found in the archive.

Expression:

```
/downloads/userguide.zip/#archive#//*.jar
```

Value: The jar files contained by the archive file.

Expression:

```
/downloads/userguide.zip/#archive#//*.jar/#archive#//*.properties/file-content()
```

Value: The content of all .properties files contained by .jar files contained by the archive file. This examples demonstrates access to nested archive files.

Navigation of BaseX databases

You can navigate BaseX databases. Use the URI scheme `basex:/`, followed by a second slash representing the root of the BaseX file system.

Examples

Expression:

```
basex://*
```

Value: All BaseX databases which are accessible from the standalone client.

Expression:

```
basex://u*
```

Value: All BaseX databases with a name starting with “u”.

Expression:

```
basex://*[.//*.xsd]
```

Value: All BaseX databases containing XSDs.

Expression:

```
basex://*//*.xsd/..
```

Value: All folders in BaseX databases which contain XSDs.

Navigation of SVN repositories

You can navigate SVN repositories located in your file system or accessible via http or https. To do so, use URIs consisting of two parts:

- Location of the repository, preceded by “svn-“
- Steps navigating the repository

Examples

Several examples use the public repository found at this address:

https://svn.apache.org/repos/asf

Expression:

```
svn-https://svn.apache.org/repos/asf/*
```

Value: List of all project folders. (Note that the returned URIs have a prefix “svn-“.)

Expression:

```
svn-https://svn.apache.org/repos/asf/xalan/java/trunk//*.xsl => xwrap('xsds', 'db')
```

Value: A document with root element “xsds”, containing all xsd documents found at the specified within-project path (java/trunk//*.xsd).

Navigation of github projects

You can navigate the contents of github projects. This requires, however, that you create a UTREE or UGRAPH description of the projects of interest. See sections “Navigation of UTREE file systems” and “Navigation of UGRAPH file systems” for further information.

IMPORTANT NOTE. The navigation of github projects – as well as the creation of UTREE or UGRAPH descriptions – requires a github token. See here for instructions how to create a token:

https://github.com/blog/1509-personal-api-tokens

Having received the token, you must make it accessible to the scripts `fox` and `lifis`:

(1) Store the token in a file (naming proposal: `github-token`)
(2) Edit the `github-token-location` files in folders `bin` and `lifis`: the line without preceding # must contain the fully qualified file name of your token file

Example:

If the token file is `github/token`, the `github-token-location` file should look similar to this:

```
# Fully qualified name of a file containing the github token.
# Unless the default value "/github/token" is appropriate,
# you must edit the following line (or you won't have access
# to github resources).
/github/token
```

Navigation of UTREE file systems

You can navigate literal file systems described by UTREE documents. (See lifis-tool-introduction.pdf for information about UTREE systems.) To do so, specify the folder(s) containing UTREE documents using the `fox` option –t:

-t whitespace-separated-list-of-folders-containing-utree-documents

Examples

In order to create a literal file system of type UTREE, we use the `lifis` tool (see: lifis-tool-introduction.pdf). The following call produces a UTREE system of all github projects created by organisation `oxygenxml`:

```
lifis "github?format=utree,org=oxygenxml" > /utree/oxygenxml/utree-oxygen.xml
```

When you want FOXpath to have access to the oxygenxml projects described by this UTREE system, you must pass the folder containing the describing UTREE document to the `fox` script, using option –t. In our example, the folder is `/utree/oxygenxml`.

Call:

```
fox -t /utree/oxygenxml "https://github.com/oxygenxml/*/file-name()"
```

Result: A list of all oxygenxml project names.

Call:

```
fox -t /utree/oxygenxml "https://github.com/oxygenxml/*[.//*.sch]/file-name()"
```

Result: All oxygenxml project names, considering only projects containing .sch files (schematron files).

Call:

```
fox -t /utree/oxygenxml "https://github.com/oxygenxml//*.sch\\*:pattern\@id
=> distinct-values() => sort()"
```

Result: Sorted list of all schematron pattern ids found in .sch files in oxygenxml projects.

Navigation of UGRAPH file systems

You can navigate literal file systems described by UGRAPH triple sets. (See lifis-tool-introduction.pdf for information about UTREE systems.)To do so, specify the SPARQL endpoint(s) exposing UGRAPH triple sets using the `fox` option –g:

-g whitespace-separated-list-of-sparql-endpoints-exposing-ugraph-triple-sets

Examples

The examples assume that you store the UGRAPH triples in a TDB triple store and expose them as SPARQL endpoint using a Fuseki server. Preparations:

(1) Install the RDF database TDB from here: https://jena.apache.org/
(2) Install the RDF server Fuseki from here: https://jena.apache.org/

In order to create a literal file system of type UGRAPH, we use the `lifis` tool (see: lifis-tool-introduction.pdf). The following call produces a UGRAPH system of all github projects created by organisation `oxygenxml`:

```
lifis "github?format=ugraph,org=oxygenxml" > /ugraph/oxygenxml/ugraph-oxygen.ttl
```

The tool delivers the triples as a Turtle file. Now load the tiples into a TDB triplestore:

```
tdbloader --loc=/tdb/ugraph-oxygenxml /ugraph/oxygenxml/ugraph-oxygenxml.ttl
```

In order to publish the triplestore as a SPARQL endpoint, start the Fuseki server, using the TDB triple store as data source:

```
fuseki-server --loc=/tdb/ugraph-oxygenxml /oxygenxml
```

When you want FOXpath to have access to the oxygenxml projects contained by this UGRAPH system, you must pass the SPARQL endpoint containing the describing UGRAPH data to the `fox` script, using option –g.

Call:

```
fox -g http://localhost:3030/oxygenxml "https://github.com/oxygenxml/*/file-name()"
```

Result: A list of all oxygenxml project names.

For further examples see section “Navigation of UTREE file systems”: in each example, replace optiona –t by

–g http://localhost:3030/oxygenxml

as shown in the preceding example.