The navigation of BaseX databases

Principles

The resource navigation of FOXpath cannot only be applied to the physical file system, but also to the virtual file system exposed by **BaseX databases**. URIs referencing BaseX databases or their resources (files and folders) must be prefixed with `basex://`. URI pattern:

```
basex://databaseName/path/to/file/or/folder
```

For example, the URI

```
basex://yogi02/frameworks/xforms/catalog.xml
```

references a document in database `yogi02` at path `frameworks/xforms/catalog.xml`. Similarly, the URI

```
basex://yogi02/frameworks/xforms
```

references a folder in database `yogi02` at path `frameworks/xforms`.

Special cases:

```
basex://
```

references the virtual root folder whose child folders represent databases.

```
basex://foo
```

references the database `foo`.

Navigation of BaseX databases and navigation of the physical file system are governed by the same rules of syntax and semantics. For example, the expression

```
basex://yogi02/frameworks/xforms/*.xsd[file-sdate >= '2016-01-01']
```

selects all XSD documents in the database `yogi02`, found in its folder `frameworks/xforms`, with a last modification date greater or equal to 2016-01-01. Compare this with an expression accomplishing an equivalent navigation of the physical file system:

```
/projs//yogi02/frameworks/xforms/*.xsd[file-sdate >= '2016-01-01']
```

Examples

A few examples should convey a feeling what FOXpath navigation of the BaseX databases is like. For more examples, see `foxpath-intro.pdf`. Although the examples in that article navigate the physical file system, they can be also used as models how to navigate BaseX databases: folders and files are treated exactly the same way, no matter if they are file system or database resources.

```
basex://*
```

Value: A list of all BaseX databases.

```
basex://y*
```

Value: A list of all BaseX databases whose name starts with “y” (case insensitive).

```
basex://yogi02/*
```

Value: Top-level files and folders in the database `yogi02`.

```
basex://yogi02//*[is-dir()]
```

Value: All folders in the database `yogi02`.

```
basex://yogi02//*[is-file()][file-sdate() lt '2016-11-07']
```

Value: All files in the database `yogi02`, which have a last modification date less than “2016-11-07”.

```
basex://yogi02//*.xsd
```

Value: All XSDs in the database `yogi02`.

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

Value: All XSDs in databases whose name starts with “yo” (case insensitive).

```
basex://yogi02//frameworks//*.xsd
```

Value: All XSDs in database `yogi02` which are directly or indirectly contained by folder `frameworks`.

```
basex://yogi02//frameworks/(mathml, ooxml)//*.xsd
```

Value: Alls XSDs in database `yogi02` which are directly or indirectly contained by framework `mathml` or `ooxml`.

```
basex://yogi02//frameworks/mathml/(* except *2.0)//*.xsd
```

Value: Alls XSDs in database `yogi02` which are directly or indirectly contained by a child folder of framework `mathml`, excluding the contents of child folders with a name ending with `2.0`.

```
basex://yogi02//mathml//*.xsd[contains(\*\@targetNamespace, 'MathML')]
```

Value: Alls XSDs in database `yogi02` which are directly or indirectly contained by a folder `mathml` and which have a target namespace containing the string “MathML”. This example shows that the navigation of folders and files can use predicates navigating file contents. (Note the backslashes denoting node tree navigation, rather than file system navigation.)

```
basex://yogi02//mathml//*.xsd\*\@targetNamespace => distinct-values() => sort()
```

Value: A sorted list of all target namespaces used by XSDs in database `yogi02`, in or under the folder `mathml`. This example shows that the navigation of folders and files can be continued by navigation into the node trees located by the preceding steps.

For more examples, see article `foxpath-intro.pdf`. Although those examples navigate the physical file system, rather than BaseX databases, there is no difference between file system and database navigation.

Tip – creation and use of database document catalogs

A selection of database documents can be translated into a **catalog document** (a “dcat”). XQuery code may use such a catalog in order to accomplish aggregated access to all selected documents. The creation of a catalog document is provided by FOXpath function `dcat`.

Example: the expression

```
basex://yogi02//frameworks//*.sch => dcat()
```

creates a catalog referencing all schematron documents in database `yogi02` found in or under the `frameworks` folder. The catalog looks like this:

```
<dcat targetFormat="xml" count="61"
      t="2016-11-07T23:37:48.073+01:00"
      onlyDocAvailable="false">
  <doc href="yogi02/frameworks/dita/DITA-OT/plugins/org.dita-ng.doctypes/checkShell.sch"/>
  <doc href="yogi02/frameworks/dita/DITA-OT2.x/plugins/org.dita-ng.doctypes/checkShell.sch"/>
  <doc href="yogi02/frameworks/dita/resources/dita-1.2-for-xslt2-links-checker.sch"/>
   …
</dcat>
```

XQuery code accessing the complete set of referenced documents is succinct:

```
…
let $docs := doc('dcat.xml')//@href/doc(.)
…
```

The writing of `dcat` documents is an example of how FOXpath may support conventional XQuery applications, as well as XSLT and XPROC applications.