Catmandu Tutorial

(version rev0)

Jakob Voß

Contents

1 Introduction

Catmandu is a data processing toolkit developed as part of the LibreCat project. LibreCat is an open collaboration of the university libraries of Lund, Ghent, and Bielefeld. See the project's website at http://librecat.org/ for more information.

Figure 1.1: the LibreCat logo

Although Catmandu is implemented in the Perl programming language, one can make use of it without knowing Perl. The framework is shipped with a command line client and it contains a simple domain-specific language.

2 First steps

2.1 The command line client

The command line client `catmandu` is installed together with the Perl module `Catmandu`. Calling it without any arguments list the available commands:

```
$ catmandu
    Available commands:

      commands: list the application's commands
          help: display a command's help screen

        config: print the Catmandu config as JSON
       convert: convert objects
         count: count the number of objects in a store
          data: store, index, search, import, export or convert objects
        delete: delete objects from a store
        export: export objects from a store
        import: import objects into a store
          move: move objects to another store
```

3 Processing data

3.1 Processing XML data

To import and export XML data there is the module Catmandu::XML. The following documentation requires at least version 0.03 of this module.

3.1.1 Importing XML

3.1.1.1 First steps

Let's start with the following XML file `input.xml`:

```
$ cat data/input.xml
<?xml version="1.0"?>
<doc>
  <id>1</id>
  <id>2</id>
  <xx>3</xx>
  <id>4</id>
</doc>
```

A simple conversion maps child elements of the root to hash elements:

```
$ catmandu convert XML to JSON < data/input.xml
{"id":["1","2","4"],"xx":"3"}
```

Option `--root` includes the root element:

```
$ catmandu convert XML --root 1 to JSON < data/input.xml
{"doc":{"id":["1","2","4"],"xx":"3"}}
```

To only convert selected XML elements, use option `--path`:

```
$ catmandu convert XML --path '/*/id' to JSON < data/input.xml
{"id":"1"}
{"id":"2"}
{"id":"4"}
```

By using option `--path` you virtually select new root elements (element `id` in this example), so `--path` also enables `--root` by default. You can still disable root elements or choose another name:

```
$ catmandu convert XML --path '/doc/*' --root a to JSON < data/input.xml
{"a":"1"}
{"a":"2"}
{"a":"3"}
{"a":"4"}
```

The default XML importer includes both, XML elements and XML attributes as key-value pairs:

```
$ echo '<doc x="1"><x>2</x></doc>' | catmandu convert XML to JSON
{"x":["1","2"]}
```

XML attributes can be disabled with option `--attributes`:

```
$ echo '<doc x="1"><x>2</x></doc>' | catmandu convert XML --attributes 0 to JSON
{"x":"2"}
```

3.1.1.2 Ordered XML

The default conversion is not suitable for so called "document-oriented" XML. Let's take another example in file `doc.xml`:

```
$ cat data/doc.xml
<doc>
  <title>Welcome!</title>
  <p>
    Look at my <a href="http://example.org/">homepage</a>!
  </p>
</doc>
```

The default conversion ignores mixed content text nodes and element order:

```
$ catmandu convert XML to JSON --pretty 1 < data/doc.xml
{
   "p" : {
      "a" : {
         "href" : "http://example.org/"
      }
   },
   "title" : "Welcome!"
}
```

To better support ordered XML, the option `--type` can be used to select an alternative representation of XML. The YAML exporter is used to better show the imported array structure:

```
$ catmandu convert XML --type ordered to YAML < data/doc.xml
---
- doc
- {}
- - - title
    - {}
    - - Welcome!
  - - p
    - {}
    - - "\n    Look at my "
      - - a
        - href: http://example.org/
        - - homepage
      - "!\n  "
```

With `--type ordered` each XML element is represented as ternary array with element name, attributes, and child elements.

The option `--depth` can be used to create ordered XML first starting with some nesting level:

```
$ catmandu convert XML --root doc --depth 2 to YAML < data/doc.xml
---
doc:
  p:
  - - p
    - {}
    - - "\n    Look at my "
```

```
      - - a
        - href: http://example.org/
        - - homepage
      - "!\n  "
  title:
  - - title
    - {}
    - - Welcome!
```

The record fields `doc.p` and `doc.title` at level 2 both contain an array of elements just like returned with `--type ordered`. We can use this structure to access and modify particular fields:

```
$ catmandu convert XML --root doc --depth 2 --fix 'move_field("doc.title.0.2","doc.title
---
doc:
  p:
  - - p
    - {}
    - - "\n    Look at my "
      - - a
        - href: http://example.org/
        - - homepage
      - "!\n  "
  title:
  - Welcome!
```

3.1.2 Exporting XML

An XML exporter has not been implemented yet.

3.1.3 Specific XML formats

The general XML importer and exporter can be difficult to use for more complex XML formats. For this reason there are specialized importers and exporters for particular XML-based formats.

...e.g. SRU, MODS, PICAXML...

4 Processing PICA data

PICA+ is an internal data format of legacy library automation systems produced by the Dutch company PICA (now aquired by OCLC). The module Catmandu::PICA provides Catmandu importer, exporter and fixes to work with PICA data.

5 Acknowledgement

...