PDF Bookmarks

Taco Hoekwater

1. Introduction

"A PDF document may optionally display a document outline on the screen, allowing the user to navigate interactively from one part of the document to another. The outline consists of a tree-structured hierarchy of outline items (sometimes called bookmarks), which serve as a visual table of contents to display the document's structure to the user. The user can interactively open and close individual items by clicking them with the mouse."

PDF reference manual

Not all PDF readers support document outlines (or bookmarks, as ConTEXt calls them), but quite a few do. Typically, the bookmarks open in a separate pane, and can be used to jump quickly to a particular section of the document. ConTEXt internally represents PDF bookmarks as a lua table with the following relevant entries:

`title`	`<string>`	the name of this bookmark, to be displayed in the outline pane
`name`	`<string>`	the name of the structure section used to create this bookmark
`level`	`<number>`	the nesting level of this bookmark
`realpage`	`<number>`	the page this bookmark links to
`opened`	`<boolean>`	whether this bookmark should initially be displayed opened or closed
`reference`	`<hash>`	the reference that this bookmark links to

the are some other entries, but these are the important ones.

2. Use in ConTEXt

To use bookmarks, some interaction features have to be turned on, if your document does not so already:

```
\setupinteraction
  [state=start]

\setupinteractionscreen
  [option=bookmark]

\placebookmarks
  [chapter,section]
  [chapter]
```

Assuming the PDF reader supports bookmarks, this will open the document with the bookmarks pane open and containing bookmarks for chapters and sections. The second argument to `\placebookmarks` ensures that the bookmark entries will all be visible on the initial document opening (read as: and open up the chapter bookmarks).
Note that the unnumbered headings (`\title`, `\subject` etc.) do not create bookmarks, even when listed in `\placebookmarks`. In order to create the bookmarks, you need to add `force=yes`, for example like this:

```
\placebookmarks
  [title, subject]
  [force=yes]
```

With a long heading in a huge font, you might want to add linebreaks by hand. No problem, just use `\\`. The bookmark code ignores `\\`, so the bookmark itself won't have a linebreak. For example:

```
\startchapter[title=Long\\ title]
hello
\stopchapter
```

ConTEXt attempts to replace commands inside bookmarks with an acceptable string. However, the result is not always optimal. To tweak ConTEXts behaviour, add specific commands to `\simplifiedcommands`.
For example, to replace the `\CONTEXT` logo (which would normally become CONTEXT) with a camel-cased version, use the following:

```
\appendtoks
    \def\CONTEXT{ConTeXt}
\to \simplifiedcommands
```

A more general method, also usable for the above problem, is to use the bookmark option to specify the bookmark text explicitly. For example:

```
\startchapter[title=A long chapter\\
                    about splines,
              bookmark=Splines]
hello
\stopchapter
```

If you do not like seeing the structure numbering in the bookmarks, you can add number=no, like this:

```
\placebookmarks
  [chapter,section]
  [chapter]
  [number=no]
```

When including pages from an external PDF document, sometimes it is useful to import the bookmarks from that PDF file as well. Doing this is very simple:

```
\externalfigure
   [externalpdf]
   [page=1,interaction=all]
```

In real life, you probably want to include more pages, but just a single page `interaction=all` is enough for ConTEXt to read all of the external bookmarks and store them in a Lua table.
ConTEXt will then automatically merge the bookmarks for the relevant pages below the current section level and patch the page numbers. It also creates a completely new bookmark as root entry for the external bookmarks. This new bookmark has the filename of the external figure as its title, and it points to the first included page.

3. Advanced processing

If you are unhapppy with ConTEXt's way of adding bookmark items, you can intercept the bookmark creation process, using some Lua code. You are allowed to hook into the normal ConTEXt processing at two places: just after ConTEXt has collected all the raw bookmarks from your document into a table, and again after all ConTEXt's automatic processing is complete. The first entry point is useful for example to get the external bookmarks and store them in a safe spot, so you can do specialized processing later on.

Here is a bit of code:

```
\startluacode
userdata.bookmarks = {}

local xtr = structures.bookmarks.extras
local premerge = function(levels)
   for _,v in ipairs(xtr.get()) do
      userdata.bookmarks[v.name]
                            = v.levels
   end
   xtr.reset()
   return levels
end

structures.bookmarks.installhandler
  ("check before","before", premerge)
\stopluacode
```

The `levels` argument to the function is the current set of bookmarks (remember, bookmarks are an array, there is no nesting). But in this example we are ignoring that argument.
The function `premerge` stores *all* the found external bookmarks (even those for pages you do not include) in your own `userdata.bookmarks` table. The command `xtr.get)` returns a table with `name`, `levels` pairs. The `v.name` entry is the name of the external file, and `v.levels` contains all the bookmarks from that file.
The `xtr.reset()` call makes sure that ConTEXt does not process the external bookmarks itself. It will still create the extra bookmark with the external file's title, which is useful if you want to merge bookmarks in manually.

The second entry point comes after all of ConTEXt's processing:

```
\startluacode
local function merge (levels)
  --   table.print(levels)
  local refs
  local collect = {}
  for _,v in ipairs (levels) do
    collect[#collect+1] = v
    refs = userdata.bookmarks[v.title]
    if refs and #refs>0 then
      for _, i in ipairs (refs) do
        i.level = i.level + v.level+1
        i.section = v.section
        collect[#collect+1] = i
      end
    end
  end
  return collect
end

structures.bookmarks.installhandler
  ("merge","after", merge)
\stopluacode
```

In this second part, you can alter `levels` in any way that you see fit. You could even delete some of the 'regular' bookmarks, or alter their content. In this case, I am inserting all of the external bookmarks, even the ones for pages that are not included.
To see what all is in `levels`, it may be useful to uncomment the `table.print()` line.
After this `"after"` step, there is nothing more that ConTEXt does with the bookmarks except write them out to the PDF document.