in-dexter

An index package for Typst

Version 0.7.0 (6.12.2024)

Rolf Bremer, Jutta Klebe

Contributors: @epsilonhalbe, @jewelpit, @sbatial, @lukasjuhrich, @ThePuzzlemaker, @hurzl

Content

1 Sample Document to Demonstrate the in-dexter package

This document explains how to use the `in-dexter` package in typst. It contains several samples of how to use `in-dexter` to effectively index a document. Make sure to look up the typst code of this document to explore, what the package can do.

Using the in-dexter package in a typst document consists of some simple steps:

1. Importing the package `in-dexter`.
2. Marking the words or phrases to include in an index.
3. Generating the index page(s) by calling the `make-index()` function.

1.1 Importing the Package

The in-dexter package is currently available on GitHub in its home repository (https://github.com/RolfBremer/in-dexter). It is still in development and may have breaking changes in its next iteration.

```
#import "./in-dexter.typ": *
```

The package is also available via Typst's built-in Package Manager:

```
#import "@preview/in-dexter:0.7.0": *
```

Note, that the version number of the typst package has to be adapted to get the wanted version. It may take some time for a new version to appear in the typst universe after it is available on GitHub.

1.2 Marking of Entries

We have marked several words to be included in an index page. The markup for the entry stays invisible. Its location in the text gets recorded, and later it is shown as a page reference in the index page.

```
#index[The Entry Phrase]
```

or

```
#index([The Entry Phrase])
```

or

```
#index("The Entry Phrase")
```

Entries marked this way are going to the "Default" Index. If only one index is needed, this is the only way needed to mark entries. In-dexter can support multiple Indexes . To specify the target index for a marking, the index must be addressed.

```
#index(index: "Secondary")[The Entry Phrase]
```

This is the explicit addressing of the secondary index. It may be useful to define a function for the alternate index, to avoid the explicitness:

```
#let index2 = index.with(index:"Secondary")

#index2[One Entry Phrase]
#index2[Another Entry Phrase]
```

1.2.1 Nested entries

Entries can be nested. The `index` function takes multiple arguments - one for each nesting level.

```
#index("Sample", "medical", "tissue")
#index("Sample", "musical", "piano")
#index("Sample")
```

1.2.1.1 LaTeX Style index grouping

Alternatively or complementing to the grouping syntax above, the "bang style" syntax known from LaTeX can be used:

```
#index("Sample!medical!X-Ray")
```

They can even be combined:

```
#index("CombiGroup", "Sample!musical!Chess!")
```

Note that the last bang is not handled as a separator, but is part of the entry. To use the bang grouping syntax, the `make-index()` function must be called with the parameter `use-bang-grouping: true`:

```
#make-index(use-bang-grouping: true)
```

1.2.2 Entries with display

These entries use an explicit display parameter. It is used to display the entry on the index page. It can contain rich content, like math expressions:

```
#index(display: "Level3", "Aaa-set3!l2!l3")
#indexMath(display: [$cal(T)_n$-set], "Aa-set")
#indexMath(display: [$cal(T)^n$-set], "Aa-set4")
```

Note that display may be ignored, if entries with the same entry key are defined beforehand. The first occurrence of an entry defines the display of all other entries with that entry key.

1.2.3 Advanced entries

Simple math expressions can be used as entry key, like the following sample, where we also provide an initial parameter to put sort the entry under “t” in the index:

```
#indexMath(initial: "t")[$t$-tuple]
```

but note, that more complex ones may not be convertible to a string by in-dexter. In such cases it is recommended to use the display parameter instead:

```
#indexMath(initial: "t", display: [$cal(T)_n$-c], "Tnc")
```

this will put the entry in the “t” section, and uses the key (“Tnc”) as sort key within that ‘t’ section. The entry is displayed as `$cal(T)_n$`.

The following entry will place the entry in the “D” section, because we have not provided an explicit `initial` parameter, so the section is derived from the keys first letter (“DTN”).

```
#indexMath(display: [d-$cal(T)_n$], "DTN")
```

The index-function to mark entries also accepts a tuple value:

```
#indexMath(([d-$rho_x$], "RTX"))
```

The first value of the tuple is interpreted as the `display`, the second as the `key` parameter.

1.2.3.1 Suppressing the casing for formulas

Sometimes, the entry-casing of the `make-index()` function should not apply to an entry. This is often the case for math formulas. The `index()` function therefore has a parameter `apply-casing`, that allows to suppress the application of the entry-casing function for this specific entry.

```
#index(display: $(n, k)"-representable"$, "nkrepresentable", apply-casing: false)
```

Note: If multiple entries have the same key, but different apply-casing flags, the first one wins.

```
#index(display: $(x, p)"-double"$, "xprepresentable", apply-casing: false)
#index(display: $(x, p)"-double"$, "xprepresentable", apply-casing: true)
```

1.2.3.2 Symbols

Symbols can be indexed to be sorted under `"Symbols"`, and be sorted at the top of the index like this

```
#indexMath(initial: (letter: "Symbols", sort-by: "#"), [$(sigma)$])
```

1.2.3.3 Formatting Entries

Entries can be formatted with arbitrary functions that map `content` to `content`

```
#index(fmt: it => strong(it), [The Entry Phrase])
```

or shorter

```
#index(fmt: strong, [The Entry Phrase])
```

For convenience in-dexter exposes `index-main` which formats the entry bold. It is semantically named to decouple the markup from the actual style. One can decide to have the main entries slanted or color formatted, which makes it clear that the style should not be part of the function name in markup. Naming markup functions according to their purpose (semantically) also eases the burden on the author, because she must not remember the currently valid styles for her intent.

Another reason to use semantically markup functions is to have them defined in a central place. Changing the style becomes very easy this way.

```
#index-main[The Entry Phrase]
```

It is predefined in in-dexter like this:

```
#let index-main = index.with(fmt: strong)
```

Here we define another semantical index marker, which adds an “ff.” to the page number.

```
#let index-ff = index.with(fmt: it => [#it _ff._])
```

1.2.3.4 Referencing Ranges and Continuations

Up to this point, we used Cardinal Markers to mark the index entries. They are referred to with their single page number from the index page. But `in-dexter` also supports more advanced references to marked entries, like the following:

- Ranges of Pages:
  - 42-46
  - 42-46, 52-59
- Single Page Continuation (SPC):
  - 77f.
  - 77+
- Multi-Page Continuation (MPC):
  - 82ff.
  - 77-
  - 77++

The Continuation Symbols (“f.”, “ff.”) symbols can be customized via parameters `spc` and `mpc` to `make-index()`.

This Sample uses “+” for *Single Page Continuation* and “++” for *Multi Page Continuations*.

```
#make-index(
  use-bang-grouping: true,
  use-page-counter: true,
  sort-order: upper,
  spc: "+",
  mpc: "++",
)
```

If `spc` is set to `none`, an explicit numeric range is used, like “42-43”. If `mpc` is set to `none`, an explicit numeric range is used, like “42-49”.

Note that “f.” and “ff.” are the default symbols for `scp` and `mcp`.

1.2.3.4.1 Range of Pages

To mark a Range of pages for an index entry, one can use the following marker:

```
#index(index-type: indextype.Start)[Entry]

// other content here

#index(index-type: indextype.End)[Entry]
```

Of course, you can shorten this somewhat explicit marker with your own marker, like this:

Behavior:

- If the markers are on the same resulting page, they are automatically combined to a Cardinal Marker in the generated index page.
- If the End-Marker is on the next page following the Start-Marker, the Marker is handled as a Continuation Marker (“f.”). If it uses the Continuation Symbol or the page numbers can be configured in a Parameter of `make-index()`.
- If there is a Start-Marker, but no End-Marker, the Marker is handled as a Continuation Marker (“ff.”).

1.3 The Index Page

To actually create the index page, the `make-index()` function has to be called. Of course, it can be embedded into an appropriately formatted environment, like this:

```
#columns(3)[
    #make-index()
]
```

The `make-index()` function accepts an optional array of indexes to include into the index page. The default value, `auto`, takes all entries from all indexes. The following sample only uses the entries of the secondary and tertiary index. See sample output in Section 3.4.

```
#columns(3)[
    #make-index(indexes: ("Secondary", "Tertiary"))
]
```

1.3.1 Skipping physical pages

If page number 1 is not the first physical page of the document, the parameter `use-page-counter` of the `make-index()` function can be set to `true`. Default is `false`. In-dexter uses the page counter instead of the physical page number then.

2 Why Having an Index in Times of Search Functionality?

A *hand-picked* or *handcrafted* Index in times of search functionality seems a bit old-fashioned at the first glance. But such an index allows the author to direct the reader, who is looking for a specific topic (using index-main), to exactly the right places.

Especially in larger documents and books this becomes very useful, since search engines may provide too many locations of specific words. The index is much more comprehensive, assuming that the author has its content selected well. Authors know best where a certain topic is explained thoroughly or merely noteworthy mentioned (using the `index` function).

Note, that this document is not necessarily a good example of the index. Here we just need to have as many index entries as possible to demonstrate (using a custom made `index-ff` function) the functionality and have a properly filled index at the end.

Even for symbols like `(ρ)`. Indexing should work for for any Unicode string like Cyrillic (Скороспелка) or German (Ölrückstoßabdämpfung). - though we need to add initials:

`#index(initial: (letter: "C", sort-by: "Ss"), "Скороспелка")`

or

`#index(initial: (letter: "Ö", sort-by: "Oo"), "Ölrückstoßabdämpfung").`

---

3 Index pages

In this chapter we emit several index pages for this document. We also switched page numbering to roman numbers, to demonstrate in-dexters ability to display them, if the option `use-page-counter` has been set to true.

3.1 The Default Index page

Here we generate the Index page in three columns. The default behavior (auto) is to use all indexes together.

3.2 Secondary Index

Here we select explicitly only the secondary index.

E

Example	5

S

Specific	5

3.3 Tertiary Index

Here we select explicitly only the tertiary index.

E

Engines	5

F

Filled	5

3.4 Combined Index

Here we select explicitly secondary and tertiary indexes.

E

Engines	5
Example	5

F

Filled	5

S

Specific	5

3.5 Combined Index - all lower case

Here we select explicitly secondary and tertiary indexes and format them all lower case.

E

engines	5
example	5

F

filled	5

S

specific	5

3.6 Math Index

Here we explicitly select only the Math index.

SYMBOLS

A

D

R

T