in-dexter

An index package for Typst

Version 0.0.6 (30. September 2023)

Rolf Bremer, Jutta Klebe

1 Sample Document to Demonstrate the in-dexter package

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 the index.
3. Generating the index page 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 build-in Package Manager:

```
#import "@preview/in-dexter:0.0.6": *
```

Note, that the version number of the typst package has to be adapted to get the wanted version.

1.2 Marking of Entries

We have marked several words to be included in an index page at the end of the document. 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]
```

1.2.1 Marker Classes

The entries support a class. This class determines the visualization for the page number of the entry. Currently, we distinguish between class "simple" and class "main" . The first one is the default. The second is provided to mark the main reference for that entry – its page number will be printed in **bold**.

```
#index(class: classes.main)[The Entry Phrase]
```

In future versions of this package there may be more marker classes for additional cases. It is recommended to use the `classes` definition of the package.

- `classes.simple`
- `classes.main`

1.2.1.1 More Convenience

There is also a convenience function, to ease the usage of main entries. Instead of the main entry syntax used above, one can use the following:

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

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()
]
```

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, 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 specific topic is explained thoroughly (using the `index-main` function to point there) 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 the functionality and have a properly filled index at the end.

---

3 Index

Here we generate the Index page in three columns:

A

Authorsresponsibility 2

B

Books 2

Breaking Changes 1

C

Classes **1**

Comprehensive 2

Content 2

Convenience **1**

D

Demonstrate **2**

Development 1

E

Entries 2

Environment 2

Explained 2

F

Formatting 2

Functionality 2

H

Hand Picked 2

Handcrafted 2

I

Index 2

Index Page 1, **2**

Invisible 1

Iteration 1

L

Large Documents 2

M

Main 1

Marker Classes **1**

N

Noteworthy 2

O

Old-fashioned 2

P

Properly 2

Provide 2

S

Sample 1

Search Engines 2

Search Functionality 2

Searching vs. Index **2**

Simple 1

T

Thoroughly 2

Topic 2