How to generate the documentation for the or-tools library

Nikolaj van Omme[*]

May 16, 2012

Abstract

This little document explains how to generate and upload all the documentation for the Google or-tools library. This document doesn't explain how to write the documentation i.e. the use of *Sphinx*, *Jinja2*, *html*, *css*, etc., only how to generate the output files once the documentation is written and how to upload the output files on the Google servers. All the scripts are written in Python and must be called in the correct sequence.

Only tested under Linux.

Contents

1 Introduction (How it works) **2**
1.1 Vocabulary . . . 2
1.2 Directories . . . 2
1.3 Work flow . . . 3
1.4 Preferred use . . . 4

2 Install the documentation and the needed tools **4**
2.1 External libraries and tools . . . 4
2.2 Scripts . . . 4

3 Update the documentation sources (SVN) **4**

4 Write the documentation OK **4**
4.1 To add a chapter . . . 5
4.2 To add a section . . . 5
4.3 To add a label and a reference . . . 5
4.4 Front material . . . 5

5 Generate the documentation **6**
5.1 The manual . . . 6
5.2 The documentation hub . . . 6
5.3 The tutorial code . . . 6
5.4 The slides . . . 6

6 Deploy the documentation **6**

7 Check the documentation **6**

8 Update the version **6**

9 Commit your changes in the source files **6**

10 Prepare the documentation **6**

[*]You can reach me at ortools.doc@gmail.com if you need some help. ;-)

11 Upload the documentation **6**
11.1 The version 6
11.2 Documentation Hub 6

12 Global cripts **7**

13 Gotchas **7**

14 Resources **7**
14.1 Images 7

1 Introduction (How it works)

This documentation and the process to generate the or-tools library's documentation was designed with the or-tools people in mind. As such, the whole generation process isn't bullet-proof nor idiot-proof.

You can get your local copy of the documentation and generate it if you want but there is no reason to do so as all the generated documentation is already available on the Google servers. If you find better way to generate some parts of the documentation, please share with the whole community (but keep in mind that our main purpose is not to spend too much time on this).

This document itself is written in LaTeX and generated with PDFLaTeX.

1.1 Vocabulary

To avoid misunderstanding, let's agree on some wordings.

Source/Code file a file in wich you write the documentation. It can be restructuredText (.rst), LaTeX (.tex) or text (.txt, .css, html, . . .)[1]. Some files are at the same time source and output files.

Output file a file that will be published on the server side, including html files, images, etc. Some files are at the same time source and output files.

Local home the computer you use to write the documentation.

Google servers the computers where the documentation is publicly accessible from the Internet.

1.2 Directories

There are five main directories: three on the Local home side and two on the Google servers side. Figure 1 illustrates those directories. The three local directories can be named to your liking (see section 2).

`SOURCES` This local directory contains the sources files, the scripts files and some configuration files. This directory is a local copy of the the directory `doc_sources`.

`DEPLOY` This local directory contains all the generated output files. It is not a local copy of the `trunk/documentation` directory as the directory structure is slightly different.

`DOCUMENTATION` This local directory is a local copy of the `trunk/documentation` directory.

`svn/doc_sources` A server directory containing the sources files of the documentation. All the files you need to generate the documentation are stored in this directory. The full url is http://or-tools.googlecode.com/svn/doc_sources.

`files` A server directory containing the downloadable files. The full url is http://or-tools.googlecode.com/files. Notice that you don't have public access to the directory, only to the files stored in it.

`svn/trunk/documentation` A server directory with the or-tools library's documentation (except for the downloadable files). The full url is http://or-tools.googlecode.com/svn/trunk/documentation.

[1]Basically, *all* source files are text files.

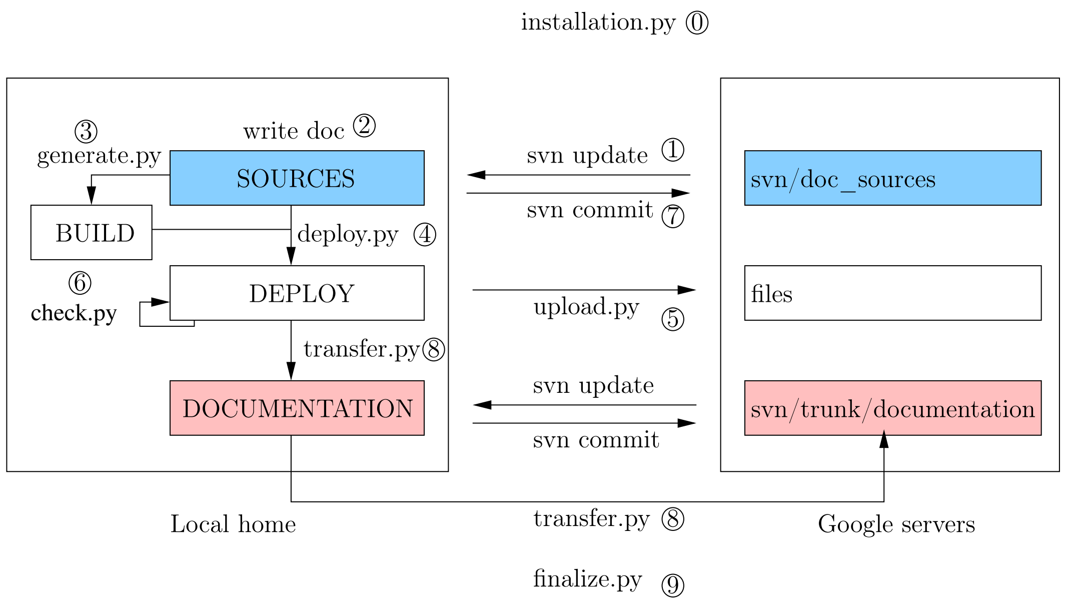

Figure 1: The six main directories and the scripts/commands to move the files between them. The base url for the Google Servers is http://or-tools.googlecode.com/.

1.3 Work flow

The whole process is divided in 11 steps[2]. Most of the steps are automated by Python scripts and to simplify your life (and mine), several scripts are bundled in the global scripts (see section 12). These scripts need to be called in sequence.

The whole process is divided in eight steps (see Figure 1) you have to follow in sequence.

Step 1 Update the source code: (script: none)
Update your local copy of the documentation via svn. See section 3.

Step 2 Generate the documentation: (script: `generate.py`)
Use Sphinx to generate the documentation. See section 5.

Step 3 Deploy the documentation: (script: `deploy.py`)
Copy locally the generated output files in the directory DEPLOY. See section 6 to understand why.

Step 4 Check the documentation: (script: `check.py`)
Check the documentation without messing with the SVN tree. See section 7.

Step 5 Update the version: (script: `version.py`)
If you're happy with the new documentation, update the new version number and keep the old one. See section 8.

Step 6 Commit your changes in the source files: (script: none)
. Commit your changes. See section 9.

Step 7 Prepare the documentation: (script: `prepare.py`)
Prepare the documentation to be uploaded. See section 10.

Step 8 Upload the documentation: (script: `upload.py`)
Upload the generated output files on the Google servers. See section 11.

[2]Once you get used to the documentation generation, you will be able to skip some steps (when you don't need to update the whole documentation tree) and to only use the specific manual commands you need.

1.4 Preferred use

These scripts are not bullet-proof nor idiot-proof. I strongly suggest that the source files always match the documentation files in http://or-tools.googlecode.com/svn/trunk/documentation . Once you have finished steps ① - ⑤, commit your changes in the source files. If there is a conflict, start the whole process again later when the generated documentation is up to date and matches the sources files on the Google servers. One easy way to ensure the consistency between the source and the documentation is to delegate to one single person the upload and commitment of the documenation (steps ⑦ - ⑪) (or to redesign the whole process ;-)), i.e. to allow only one person to perform steps ⑦ - ⑪. This design ask for some discipline but I think it is manageable among Googlers.

2 Install the documentation and the needed tools

2.1 External libraries and tools

The following list details all the libraries and tools that I use to generate the documentation. Most of them are written in Python.

Sphinx This is the main library. It transforms `restucturedText` into plenty of other formats.

Jinja2

LaTeX To generate the pdf version of the manual, we use LaTeX and `pdflatex` (and of course, the classical `makeindex`/`mkindex`, `bibtex`, . . .).

We use also Python (2.7), make, html, css, ...

2.2 Scripts

The Python scripts are not bullet-proof.

There are three main directories:

SOURCES contains all the sources. This is the directory where you write the documentation. This directory is located in the `or-tools/documentation/SOURCES` directory of the project.

DEPLOY is an "exact" copy of the `or-tools/trunk/documentation` directory. Some directories/files are only present as links (for instance `reference_manual`).

SVN is an exact copy of the `or-tools/trunk/documentation` directory.

I use the directory `DEPLOY` as an intermediate directory to be able to test some new stuff without polluting `SVN`.

Step 1 `deploy.py` generates and copies the documentation to directory `DEPLOY`. It is also responsible for the generation of all needed files and repertories.

Step 2 `upload.py` uploads certain files to the `download` directory of the or-tools project and copies the documentation to directory `SVN`. Along the way, this script generates the list of `svn` commands in `snv_commands_xxx.txt` and this is the **only** file that it generates.

Step 3 `svn`: the traditional `svn` commands.

3 Update the documentation sources (SVN)

4 Write the documentation OK

This section convers only the manual. To correct or add some material, just open the corresponding `rst` file, follow the `rst` syntax and the more specialized `Sphinx` syntax and adapt the file to your needs. If you want to add a chapter or a section, read the next two sections. We discuss also how to add a label and a reference in section 4.3 and the front material as it requires a special treatment.

4.1 To add a chapter

To add a chapter, follow the next steps:

1. In `SOURCES/MANUAL/source/manual`, create a file `mychapter.rst` and a folder `mychapter` where you will write the different sections of the chapter. See the other `rst` files.
2. Add an entry in `index.rst` in the table of contents at the end. This will add you chapter to the manual but will not make it visible yet.
3. Update manually the table of contents and possibly renumber the other chapters:
   - In `index.rst`, update the sidebar (`.. sidebar:: Content at a glance`);
   - In `MANUAL/source/doctemplates`, update `myglobaltoc.html`.This file is used to generate the toc in the sidebar for the first page of each chapter in the `html` version.
4. To make you chapter visible, open `config.py` and update `html_sidebars` accordingly.

There is nothing more to do for the LaTeX version.

4.2 To add a section

Just write your section in a `rst` file in the folder corresponding to the chapter and add an entry in the corresponding `rst` file of the chapter.

4.3 To add a label and a reference

Use the `rst` syntax: `.. _myreference:`. To refer to this reference, unfortunately, you have to produce two versions: one for the `html` version and one for the LaTeX version.

Here is an example:

```
.. _myreference:

Mysection
---------

Text text text text text text ...

.. raw:: latex

    You can find more in section~\ref{manual/chaptername/filename_without_extension:myreference}
    ...

.. only:: html

    In :ref:‘Mysection <myreference>‘, we cover ... in more details.
```

This duplication is needed because references are treated differently in the `pdf` manual and the `html` version. In the `pdf` manual, you refer by the corresponding numbers (like this: You can find more in section 4.3 . . .) while in `html`, your refer with the title name and a link (like this: In To add a label and a reference, we cover . . . in more details).

See the gotchas about some references and how `Sphinx` transforms them.

4.4 Front material

Again, we have duplicated the text as `Sphinx` treats the `pdf` and `html` versions differently.

4.4.1 Title page

4.4.2 Foreword

4.4.3 Table of contents

5 Generate the documentation

5.1 The manual

5.1.1 *Final* or *draft* release

5.2 The documentation hub

5.3 The tutorial code

5.4 The slides

6 Deploy the documentation

7 Check the documentation

8 Update the version

9 Commit your changes in the source files

10 Prepare the documentation

11 Upload the documentation

11.1 The version

All the automatic generation of the doc is based on the current version number, so don't mess with it[3]! ;-) The file `current_version.txt` contains the current version (the one that you will upload on the server). You can update it by hand. By default, after a deploy and an update, the next version is incremented by 1. For instance, if the current version was `1.2.23` before deploying and updating, then the next version will be automatically set to `1.2.24`.

Each documentation of subdirectory `SOURCES` has an individual `deploy_xxx` and `upload_xxx` than can be called manually if desired. Pay attention to the order in which you call them. We detail each of them in the right sequence in the next subsections.

11.2 Documentation Hub

11.2.1 The change files/directory

changes.txt This is where you write what changed since the last upload of the documentation. This file is automatically inserted in `documentation_hub.html` so be carefull! ;-) Lines starting with `#` are comments that are not written in the html file. Don't add the version, this is done automatically. Note that this file is not automatically updated. You are responsible for its content.

changes_list.txt This file is automatically updated with the content of `changes.txt`.

changes This directory contains copies of all the `changes.txt` files.

[3] An assert-like script is run by all the other scripts to verify that at least the current version is greater than the version before.

12 Global cripts

13 Gotchas

- When in draft mode (see 5.1.1), the section numbers in the html pages are wrong.
- LaTeX slides files have to start with `\documentclass{}` on the first line (required by `generate_slides.py`).
- Manual: the preface is copied once. This is necessary as Sphinx doesn't allow to differentiate between titles in LaTeX and Sphinx.
- There is no automatic update between the Getting started page of the wiki and the one of the manual. If you change one, you have to change the other manually.
- When you want to use class names in titles and want to talk about them in the plural, you have to add `\s`, not just `s`:

  ```
  ``SearchMonitor``\s
  -------------------
  ```

- Capitalized letters in a reference become lower letters and dashes ("-") become underscores ("_") in the LaTeX version of the references. So for instance, if the reference in your `rst` file `my_file.rst` in the directory `manual/my_chapter/` is `MyStrange_referenceIa`, the LaTeX reference will be `manual/my_chapter/my_file:mystrange-referenceia`.

14 Resources

14.1 Images

Xfig

fig2pdf