How to generate the documentation for the or-tools library

Nikolaj van Omme*

May 27, 2013

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. how to 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.

THIS INTERNAL DOCUMENT IS ONLY HELPFULL IF YOU WANT TO GENERATE THE DOCUMENTATION![1]

AS OF MAY 23, 2013, THIS DOCUMENT ACCURATELY DESCRIBES THE DOCUMENTATION GENERATION PROCESS.

Please update this document if you change anything to the documentation generation process.

The 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. In agreement with Laurent Perron, some tools were disregarded.

The contract only permitted to deal with a first version of the manual with the accompanying C++ code and the documentation HUB. The FAQ, slides and other documentation haven't been done ... yet. In particular, the C++ code hasn't been translated into other languages.

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

Contents

*You can reach me at ortools.doc@gmail.com (or ask Laurent on how to reach me) if you need some help. ;-)

[1] 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.

1 Preminilaries

We quickly describe the involved directories and files.

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, . . .)[2]. 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.

Root directory The local directory in your **Local home** that contains locally all the files to generate the documentation. It is your workspace to generate the documentation.

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

Documentation hub A central html page from which all the documentation is accessible except the downloadable files. The url is:
http://or-tools.googlecode.com/svn/trunk/documentation/documentation_hub.html.

1.2 Directories

There are **seven** main directories (illustrated on Figure 1 page 6).

Four on the **Local home** side:

`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`.

`BUILD` This local directory contains all the automatically generated output files.
Exceptions: `HUB` (is build inside its source directory).

`DEPLOY` This local directory contains all the generated output files (automatically or by hand). It is not a local copy of the `trunk/documentation` directory as the directory structure is slightly different. We use it essentially for testing purpose. Consider this directory as a buffer between `BUILD` and `DOCUMENTATION`. This buffer is also handy is the documentation is created and generated concurrently.

`DOCUMENTATION` The real thing, i.e. the documentation you can access from the documentation hub. This local directory is a local copy of the `trunk/documentation` directory. Actually, it is more the opposite: `svn/trunk/documentation` is a copy of your local directory `DOCUMENTATION` as we push (commit) the generated documentation on the Google servers without pulling (updating) it.

and **three** on the **Google servers** side:

`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.

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

`files` A server directory containing the downloadable files. The full url is http://or-tools.googlecode.com/files. Note 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.

1.3 The SOURCES directory

The documentation is explained on the **Documentation hub**.

The local directory `SOURCES` contains the following files:

`current_version.txt` Text file with the actual release number (xx.yy.zz), i.e. the release number of the next documentation you will upload. It is automatically updated after a new release is uploaded but you can manually update it if necessary.

`global.rst` Rst file with global variables.

`README.txt` You know . . . the file you never read but really should read...

`TODO.txt` What we plan for the future.

The local directory `SOURCES` contains the following directories:

`doc` The doc of the doc, i.e. the LaTeX files to generate this document.

`FAQ` Source files for the Frequently Asked Questions. The FAQ is generated by Sphinx. Needs to be rewritten and completed.

`HUB` Sources files for the documentation hub. Written by hand but with the help of the Python `generate_hub.py` script.

`images` Interesting images to use somewhere else. Kept here meanwhile.

`internals` Internal files for the scripts. You shouldn't change anything here.

`LABS` Exercises. Needs to be written.

`LICENSES` Different licenses. They are all regrouped in one place to easier update them as needed.

`MANUAL` The user's manual directory.

`scripts` Several Python scripts used throughout the or-tools documentation project.

`SLIDES` Tentative try to generate slides for the documentation. Has to be (re)written and a new approach with pygment is needed.

`sphinxortools` Tentative or-tools Sphinx extension. Abandonned after too many hair pulled out (Sphinx bugs). Another avenue would be to use pre- and post-processing tools currently not recommended by Laurent. Because of numerous Sphinx (last version used: 1.1.3) bugs, you need to duplicate lots of texts, tags and tables in the source files.

`TUTORIALS` C++ code and hopefully other languages one day.

2 How to Install the documentation sources and the needed tools

We describe here how to do a fesh installation of the sources to generate the documentation.

2.1 Local directories and copies

First, you need to setup your **Local Root** directory as your working directory. Inside this directory, create the following directories:

- SOURCES
- BUILD
- DOCUMENTATION
- DEPLOY

SOURCES and DOCUMENTATION are svn copies of svn/doc_sources and svn/trunk/documentation so you have to create them with svn:

```
svn checkout https://or-tools.googlecode.com/svn/doc_source SOURCES --username me

svn checkout https://or-tools.googlecode.com/svn/trunk/documentation DOCUMENTATION
--username me
```

and type your *GoogleCode.com Password* when prompted.

BUILD and DEPLOY have to be manually created.

You also have to manually create:

- DEPLOY/UPLOAD/manual: To hold the generated manuels in pdf and epub formats.
- labs, tutorials and reference_manual: these are copies of the corresponding directories in DOCUMENTATION. The idea is to generate their content with generate_*.py and deploy_*.py scripts but this has to be done. For the moment, simply copy the directories. You don't need to copy reference_manual as a simple soft link will do.

2.2 The config.ini file

Copy the config.ini file from the SOURCES/scripts directory into your **Local Root** directory. and fill in the first lines:

```
# Configuration file for the or-tools documentation project.
# All keys MUST be in lowercase.
[root]
dir =

project = or-tools

[ortools]

dir =

[personal]
password =
username =
```

For instance, in my configuration:

- dir = /mnt/data/RESEARCH/GOOGLE/OR_TOOLS_DOC
- dir = /mnt/data/or-tools-read-only

2.3 External libraries and tools

The following list details all the libraries and tools that you need 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. The latest version the better. `Sphinx` is quite buggy but produces beautiful documents. The website can be found here:

http://sphinx-doc.org/

Some `Sphinx` extensions are used (or are planned to be used one day...):

sphinx.ext.pngmath Instead of using `MathJax`, we transform the formulas into `png`. The `MathJax` server is often too busy...

sphinx.ext.ifconfig To create different versions of the documentation (A4/USLetter, final/draft, ...).

sphinxortools.ortools Homemade extension that ... doesn't work and is abandonned.

sphinxcontrib.doxylink No used yet. To create external links with the documentation generated by Doxygen.

Jinja2 Jinja2 is a templating language for Python (used to generate the HTML code), The website can be found here:

http://jinja.pocoo.org/

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

Xfig To generate the pictures.

fig2xxx To convert `fig` files to `pdf`, `png` and `ps` when needed.

ImageMagic Very handy library to transform images.

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

2.4 Scripts and first run

By now, you should have a copy of `svn/doc_sources` in `SOURCES`, a copy of `svn/trunk/documentation` in `DOCUMENTATION` and local copies of `labs`, `tutorials` and `reference_manual`.

The first run isn't different from a usual documentation update. See section 3.1.

3 How to generate the manual

We describe here how to generate the manual. The other parts of the documentation are generated in a similar fashion. If you don't find a specific script in a subdirectory, it means that this script is not needed for that part of the documentation.

3.1 Work flow

The whole process is divided in 9 steps[3]. Most of the steps are automated by Python scripts. These scripts need to be called in sequence. In each of the subdirectories of the directory `SOURCES` (see section 1.3), you will find the corresponding Python scripts if needed.

Step ① Update the source code: (script: none)
Update your local copy of the documentation via svn: `svn update`.

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

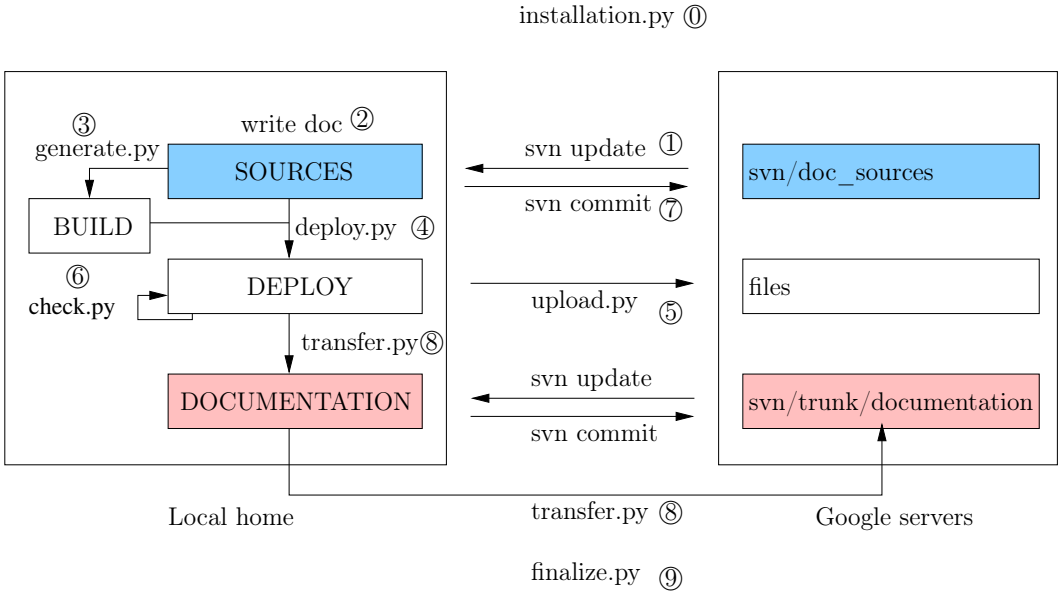

Figure 1: figure
The six main directories and the scripts/commands to move the files between them. The base url for the Google Servers is http://or-tools.googlecode.com/. The directories colored in the same color are svn copies of each other (white is not considered as a color).

Step ② Write the documentation: (script: none)
Let your talent speak!

Step ③ Generate the documentation: (script: `generate_xxx.py`)
Generate automatically the documentation. You will find the generated documentation in the `BUILD` directory. For the moment, only the manual (subdirectory `MANUAL`) and the FAQ (subdirectory `FAQ`) are automatically generated. If you are not uploading a new release on the **Google servers**, you don't need to use this script: Sphinx provides a handy `Makefile`.

The **Documentation hub** is parlty generated automatically, partly generated by hand.

Step ④ Deploy the documentation: (script: `deploy_xxx.py`)
Copy locally the output files in the directory `DEPLOY`. This directory is used to test the documentation and try new things without messing with the svn directory.

Step ⑤ Upload the documentation: (script: `upload_xxx.py`)
Before you can check the documentation in the `DEPLOY` directory, and if you want to test the download links, you have to upload the downloadable files. The `upload_xxx.py` script will be stuck if you try to upload a file that is already uploaded (based on the filename). If you want to update a downloadable file, you first have to delete the file from the Google servers. For the moment, this action has to be done manually through web user interface. If you know how to do it with a script, we are interested!

Step ⑥ Check the documentation: (script: `check.py` in the `scripts` directory)
Check the documentation. As of May 23 2013, the `check.py` script only checks internal links for the manual (via the script `check_internal_refs.py`). The rest is a manual (and important!) process (Sphinx does provide some handy tools).

Step ⑦ Update the documentation source: (script: none)
If you're happy with the new documentation, commit your changes on the Google solvers. This step can be completed after step ⑨.

Step ⑧ Upload the documentation: (script: `transfer.py` in the `scripts` directory)
Push (commit) the documentation on the Google servers. Prefer to use the `transfer.py` script over the svn commands as it completes all required tasks in a single procedure. As of 23

May 2013, `transfer.py` only calls the `svn_documentation_commit.py` script that - despite its name - only commits the html version of the manual. You can roll a dry run with the `compare_deploy_vs_documentation.py` than will create the `added.txt`, `changed.txt` and `deleted.txt` files with explanatory filenames. It's best to use this script before commiting *atrocities*.

Step ⑨ Prepare the next release of the documentation: (script: `finalize.py` in the `scripts` directory)
As of May 23 2013, `finalize.py` only calls the `versions.py` that update the version number for the next release.

The documentation has a release number (xx.yy.zz). It is used internally by the scripts to check the validity of the documentation. The `versions.py` script will copy the old number and update the new number (for instance, from 4.5.11 to 4.5.12 by default). You MUST use this script after step ⑧. If you want to update the release number to a higher release number, edit manually the file `current_version.txt` in the `SOURCES` directory WITHOUT calling this script.

3.2 Preferred use

These scripts are not bullet-proof nor idiot-proof (and can be easely improved). 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 - ⑥, commit your changes on the Google solvers (step ⑦). If there is a conflict, start the whole process again later with the updated documentation sources. One easy way to ensure the consistency between the sources and the documentation is to delegate the upload and commitment of the documentation (steps ⑧ - ⑨) to one single person (or to redesign the whole process ;-)). This design ask for some discipline but I think it is manageable among Googlers.

3.3 Different versions

The manual exists in several versions:

`a4` European `a4` format.

`letter` American `USLetter` format.

`epub` `epub` generation.

`latexpdf` `pdf` generation.

`draft` `draft` release.

`final` `final` release.

The very handy `.. only::` directive allows to target a specific version of the manual. For instance, if you don't want to publish some material, use:

```
.. only:: draft
```

You can use the `convert_draft_to_final.py` script located in the `scripts` directory to get rid of the `.. only:: draft` directives.

4 Documentation hub

Some explanation about the generation of the **Documentation hub**.

4.1 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.

5 Gotchas

Here we summarize some bugs, mistakes, difficulties, etc. When in doubt, you always can use the `restructuredText raw` directive.

- The tables are really badly handled in Sphinx...
- References are badly handled in Sphinx... You need to duplicate them in the source files if you want to generate the well-known *see section 11.4.3* for instance in LaTeX generated documents. See next item.
- Capitalized letters in a `rst` reference become lower letters and underscores ("_") become dashes ("-") in the `Sphinx` generated references. So for instance, if the reference in your `rst` file `my_file.rst` in the directory `manual/my_chapter/` is `MyStrange_referenceIa`, the `Sphinx` LaTeX reference will be `manual/my_chapter/my_file:mystrange-referenceia`.
- When in draft mode (see 3.3), 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`).
- Often, images and figures need to be duplicated: one version for the `hmtl` and `epub` manuals, one version for the LaTeX generated documents.
- 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
  -------------------
  ```