

User's Manual for `oxdoc`

Y. Zwols [`yorizwols@users.sourceforge.net`]

February 6, 2010

`oxdoc` is a small software package for generating documentation in HTML format from comments in `ox` source code. It is inspired by Sun Microsystems' Javadoc. To use `oxdoc`, the user needs to write comments in `ox` source code in a special format and run the `oxdoc` application to extract these comments and generate a comprehensive HTML (web page) document of the available functions and classes in the original code.

`oxdoc` is free software and comes with **ABSOLUTELY NO WARRANTY**. You are welcome to redistribute it under certain conditions. See the `LICENSE` file for distribution details.

Contents

1	Installation	3
1.1	Prerequisites	3
1.2	Installation	3
1.2.1	Installation on Windows 2000/XP	3
1.2.2	Installation on Linux	4
2	Running <code>oxdoc</code>	6
2.1	Running <code>oxdoc</code> using the GUI	6
2.2	Running <code>oxdoc</code> from the command-line	6
2.2.1	Options	6
3	Writing documentation	7
3.1	Adding documentation to files, classes, methods, fields	7
3.2	Documentation sections	8
3.3	Using \LaTeX -style formulas	9
3.4	Cross-referencing	10
3.5	<code>#include</code> versus <code>#import</code>	10
3.6	Documenting the internals of the project	10
4	Customizing lay-out	11
4.1	Changing the style sheet	11
4.2	Changing the HTML title	11
4.3	Adding icons	11
5	Configuration	12
5.1	Location of configuration files	12
5.2	Lay-out of <code>oxdoc.xml</code>	12
5.3	Command line configuration	12
5.3.1	\LaTeX settings	13
5.4	Overview of available settings	13
5.4.1	Options for input	13
5.4.2	Options for output	13
5.4.3	Options for third-party utilities	14

Chapter 1

Installation

1.1 Prerequisites

Since `oxdoc` was written in Java, you should have Java installed on your computer. The fact that `oxdoc` is a Java program means that it can in principle be used on any operating system, including Windows and Linux. In this section, the installation process for Windows operating systems will be described. For Linux and other operating systems, we will describe the manual installation process which is slightly more complicated.

The Java Runtime Environment (JRE) can be downloaded from www.java.com/getjava. Most Linux distributions either have Java pre-installed, or it can be installed from the installation repositories.

In order to use \LaTeX generated formulas, a copy of \LaTeX is required as well. A free Windows distribution called MiKTeX can be downloaded from <http://www.miktex.org/>. Alternatively, `oxdoc` can generate mathematical formulas using the JavaScript package MathML. Generating formulas with MathML does not require any third-party software.

`oxdoc` uses a program called `dvipng` to generate PNG (Portable Network Graphics) files from \LaTeX code. The full installation of MiKTeX comes with a version of this programs, but other distributions may not have it readily available. If you use a non-full installation of MiKTeX, make sure to select `dvipng` during the installation. Most Linux distributions have `dvipng` in their installation repositories. For example, installing `dvipng` on Ubuntu would be done by issuing the following command in a terminal:

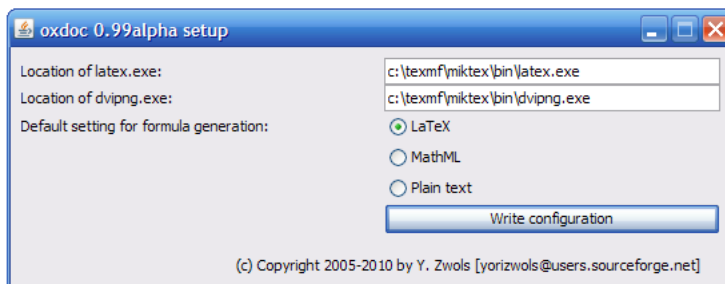
```
sudo apt-get install dvipng
```

1.2 Installation

1.2.1 Installation on Windows 2000/XP

Follow the following steps to install `oxdoc`:

1. Download the latest binary zip file (`oxdoc-xxxx-bin.zip`) from the SourceForge website (<http://oxdoc.sourceforge.net>).
2. Unzip this file in a convenient directory (for example in `C:\Program Files\oxdoc`).
3. Run the set up program. This program can be found in the subdirectory `oxdoc\bin` in the directory in which you installed `oxdoc`. Double-click on `setup.bat` to start it. You will see the following setup screen:



4. Make sure to specify the location of the `latex.exe` and `dvipng.exe` files of your MiKTeX installation (or any other \LaTeX distribution).
5. Click 'Write configuration' to save the options to the main `oxdoc` configuration file (i.e. a file named `oxdoc.xml` that is located in the 'bin' subdirectory of the `oxdoc` installation.)
6. (Optional) You may also want to add the `oxdoc` directory to your search path, so that `oxdoc` can be run from any working directory. To do so, right-click on the 'My Computer' icon on your desktop, and choose 'Properties'. Under the tab 'Advanced', click on the button 'Environment Variables'. Edit the variable 'Path' which can be found under System variables by adding a semi-colon followed by the path to the `bin` subdirectory of your `oxdoc` installation. For example, your path may now look like:

```
%systemroot%\system32;%systemroot%;C:\program files\oxdoc\bin
```

Click OK a couple of times to commit the changes. Notice that the changes will not affect only newly opened command line windows, so that all already opened command line windows will not notice the changes.

You are now ready to use `oxdoc`. In the subdirectory `oxdoc\bin` you can find the `oxdoc` graphical user interface that helps you get started with running `oxdoc`. To test whether `oxdoc` works, run the batch file `oxdoc.bat` from the command line by just typing '`oxdoc`'. It should display a short description of the program options.

1.2.2 Installation on Linux

Follow the following steps to install `oxdoc`:

1. Download the latest binary `.tar.gz` file (`oxdoc-xxxx-bin.tar.gz`) from the SourceForge website (<http://oxdoc.sourceforge.net>).
2. If you want to use \LaTeX generated formulas, make sure that you have a working version of \LaTeX and of `dvipng`. The latter is available in the standard repositories of most Linux distributions. For example, in Ubuntu, you may install `dvipng` by typing:

```
sudo apt-get install dvipng
```

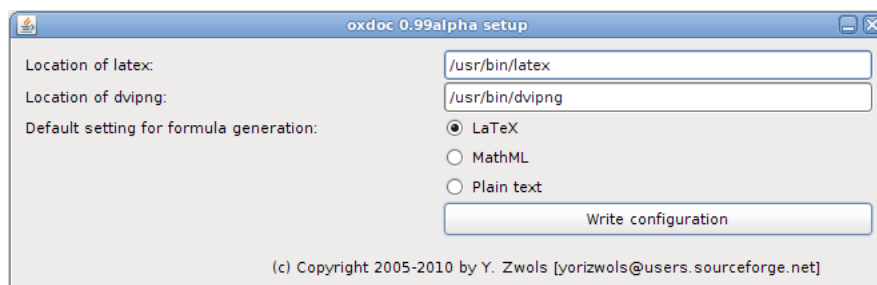
3. Extract the files from this compressed file into a convenient directory (for example in `/opt`). For example, type the following in a terminal:

```
cd /opt
tar xvfz /Desktop/oxdoc-0.99alpha-bin.tar.gz
```

4. Run the set up program. This program can be found in the subdirectory `oxdoc/bin` in the directory in which you installed `oxdoc`. So for example, type:

```
/opt/oxdoc/bin/setup
```

You will see the following setup screen:



5. Make sure to specify the location of the `latex` and `dvipng` files of your $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ installation.
6. Click 'Write configuration' to save the options to the main `oxdoc` configuration file (i.e. a file named `oxdoc.xml` that is located in the `.oxdoc` subdirectory of your home directory).

You are now ready to use `oxdoc`. In the subdirectory `bin` you can find the `oxdoc` graphical user interface that helps you get started with running `oxdoc`. You may run it by typing:

```
/opt/bin/oxdocgui
```

You may run the command line (recommended) version of `oxdoc` by typing:

```
/opt/bin/oxdoc
```

It should display a short description of the program options.

Chapter 2

Running `oxdoc`

Although `oxdoc` is a command line utility at its core, the easiest way to work with it is to use the graphical user interface (GUI). We do recommend, however, to learn how to use `oxdoc` from the command line, because this is more flexible and more convenient if you will be using `oxdoc` repeatedly. The GUI can be found in the `bin` subdirectory of your `oxdoc` installation.

2.1 Running `oxdoc` using the GUI

2.2 Running `oxdoc` from the command-line

Using `oxdoc` is rather easy. Generating documentation for an `ox` project requires running `oxdoc` and specifying the names of the files you want to generate documentation from. For example, suppose you have a number of `ox` files in a folder. From there, run

```
oxdoc *.ox
```

from the command prompt in that folder. `oxdoc` generates a set of HTML files, of which `default.html` is the project home file. It also creates a new style sheet file `oxdoc.css`.

It is advisable to specify an output directory for your project. This can be done by creating a new `oxdoc.xml` file in your project directory. See [Configuration](#) for more information on that.

2.2.1 Options

Chapter 3

Writing documentation

Now you know how to run `oxdoc`, it's time to write some comments in your code. Documentation comments consist of the normal `ox` comments, but instead of using `/*` and `*/`, we use `/**` and `**/`. The general rule is that these documentation blocks must be placed directly above the definitions and function definitions. For example:

3.1 Adding documentation to files, classes, methods, fields

Let us explain how to write documentation using `oxdoc` is by looking at the following example:

```

1  /** The multivariate Normal distribution  $\mathcal{N}(\mu, \Sigma)$ .
2  An instance of a NormalDistribution class generates realizations of a random
3  variable  $X$  with probability density function
4   $f(x) = |\Sigma|^{-1/2} (2\pi)^{-n/2}$ 
5       $\exp\left(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right)$ .
6
7  @author Y. Zwols
8
9  @example To generate 20 samples from a standard normal distribution,
10 the following code can be used:
11 <pre>
12 decl Dist = new NormalDistribution(0, 1);
13 decl Z = Dist.Generate(20);
14 </pre>
15 **/
16 class NormalDistribution {
17     /** This vector denotes the mean of this normal distribution object. */
18     decl m_vMu;
19
20     /** This matrix denotes the variance/covariance matrix of this normal distribution object. */
21     decl m_mSigma;
22
23     NormalDistribution(const vMu, const mSigma);
24     virtual Generate(const cT);
25     virtual Dim();
26 }
27
28 /** Create a new instance of the NormalDistribution class with parameters  $\mu$ 

```



```

29 and  $\Sigma$ .
30 @param vMu The mean  $\mu$  of the normal distribution
31 @param mSigma The variance/covariance matrix
32 @comments The dimension of the multivariate normal distribution is deduced
33 from the dimensions of the arguments. */
34 NormalDistribution::NormalDistribution(const vMu, const mSigma) {
35     expectMatrix("vMu", vMu, rows(vMu), 1);
36     expectMatrix("mSigma", mSigma, rows(vMu), rows(vMu));
37     m_vMu = vMu;
38     m_mSigma = mSigma;
39 }
40
41 /** Generate a vector of realizations. The length of the sample is given
42 by the argument cT.
43 @param cT Number of samples
44 */
45 NormalDistribution::Generate(const cT) {
46     return rann(cT, Dim());
47 }
48
49 /** The dimension of the multivariate normal distribution.
50 @comments This is deduced from the arguments given to the constructor. */
51 NormalDistribution::Dim() {
52     return rows(m_vMu);
53 }

```

This example shows most of the features. Every documentation block is written between `/**` and `*/` signs; HTML tags can be used, e.g. to add markup, or include images. Also, documentation blocks are divided into small sections by `@` commands. For example, parameters can be described in the `@param` section, and extra comments are given in the `@comments` section.

Also, the first sentence of the comment block is taken as a summary of the documentation block. This first sentence appears in e.g. the project home page and the methods table. `oxdoc` recognizes the first sentence by scanning for a period followed by a white space. This may have some undesired effects when a period in the first sentence doesn't indicate the end of a sentence, e.g. in the sentence

```
This class implements Dr. John's method. It solves linear equations.
```

Here, the part `This class implements Dr.` will be taken as a summary. This can be avoided by placing ` ` (a non-breaking space) just after `Dr.:`

```
This class implements Dr.&nbsp;John's method. It solves linear equations.
```

Moreover, it is possible to include any HTML tag. This may be useful for including of images, or adding more intricate mark up.

3.2 Documentation sections

There are different types of documentation sections. The following sections are available:

- `@author` specifies the author of the file. For usage, see the listing above.
- `@comments` gives comments. For usage, see the listing above.

- `@example` gives an example. For usage, see the listing above.
- `@param` describes a parameter or argument of a function. The first word after the `@param` keyword is treated as the name of the parameter. More than one parameter can be described by adding more `@param` sections.
- `@returns` describes the return value.
- `@see` gives cross references. References have to match the exact name of other entities. Multiple references have to be separated by commas.

```

1 /** Abstract distribution class
2     @see NormalDistribution, UniformDistribution */
3 class Distribution {
4     ...
5 }

```

3.3 Using \LaTeX -style formulas

Formulas can be inserted by writing them between single or double dollar (\$) signs. For example:

```

1 <pre>
2 /** Calculate the OLS estimates for the model $y = X\beta$.
3     @returns The OLS estimate $\hat{\beta} = (X'X)^{-1}X'y$.
4     */
5 regression(X, y) {
6     return invert(X'X)*X'y;
7 }
8 </pre>

```

Single dollar signs are used for inline formulas, whereas double dollar signs are used for equations on separate lines, analogously to \LaTeX . They are implemented as `align*` environments.

The way `oxdoc` processes these formulas can be changed. There are three options:

1. Plain. The formulas are copied as-is into the HTML text. This is not recommended, because it generally results in quite unreadable formulas.
2. \LaTeX . This uses the \LaTeX installation on the computer. If you didn't install `oxdoc` using the setup program, you should specify the location of `latex` and `dvipng` in `oxdoc.xml` (see also the configuration section). Choose
3. MathML. When using MathML, the formulas are rendering by a clever JavaScript program call ASCIIMathML (see <http://www1.chapman.edu/~jipsen/mathml/asciimath.html>). This script runs inside the browser and relies heavily on JavaScript. Although it gives the best-looking results, it has the drawback that it relies on the viewer's browser. It is, for example, not compatible with Google Chrome (yet?).

Alternatively, you can of course ignore `oxdoc`'s formula features and write formulas directly in HTML format by using the appropriate HTML code.

3.4 Cross-referencing

Making cross references within comments is done by placing a symbol between ‘ signs. It is important to specify the whole name of the item to be referenced. Global functions and classes are identified by their full names (this is case sensitive!) without arguments, and class methods are identified by the form `classname::method`. For example, if there is a method `isOk()` in the class `Lumberjack`, this method is referenced to by ``Lumberjack::isOk``. The same holds for the `@see` sections. Note that in `@see` sections, no ‘ signs should be used.

3.5 #include versus #import

In order to work with `oxdoc`, it may be useful to know a little bit more about how `oxdoc` works. First of all, if you have ran `oxdoc` on a project before, you may have noticed that for every input source file `filename.ox`, there is a documentation file `filename.html`. Thus, all the classes and functions that are declared in `filename.ox` and *all files that are included using the #include directive* will be documented in `filename.html`. To illustrate the implications of this, suppose that you have two files `a.ox` and `b.ox`, both defining certain classes, and two corresponding files `a.h` and `b.h`. Suppose that `a.ox` has an `#include` directive for both `a.h` and `b.h`, and that `b.ox` has an `#include` directive only for `b.h`. What will happen is that the classes defined in `a.h` and `b.h` will both end up in `a.html`, and the classes defined in `b.h` will appear a second time in `b.html`. For this reason, it is important to only `#include` the header file `filename.h` that corresponds `filename.ox`. For all other modules that you want to include, you should use `#import`. This is consistent with the suggested usage of `#include/#import` of the Ox manual.

3.6 Documenting the internals of the project

By default, `oxdoc` assumes that you are only interested in documentation about the public interface of your package. That is, it will only display public fields and methods of classes. This behavior can be reversed by specifying the `-showinternals` option on the `oxdoc` command line. This will generate documentation about *all* fields and methods.

Although `ox` supports private and protected fields, it does not support private and protected methods. However, sometimes you may want to specify that certain methods are not supposed to be called by users. This can be done by adding the `@internal` flag to their respective blocks. Methods for which the `@internal` flag is specified will be hidden from the documentation. For example:

```
1  /** Updates references. @internal **/  
2  MyClass::updateReferences()  
3  {  
4      ...  
5  }
```

Specifying the `-showinternals` option will make these methods appear again.

Chapter 4

Customizing lay-out

4.1 Changing the style sheet

A large part of the lay-out of the HTML files generated by `oxdoc` is controlled by its style sheet, `oxdoc.css`. `oxdoc` creates a default lay-out file if it is not present, but it won't overwrite changes you make to that file. The `css/` directory of your `oxdoc` installation contains a number of standard style sheet files. To choose any of them, just replace the `oxdoc.css` file by any of the files from that directory.

4.2 Changing the HTML title

Most browsers will display the title of an HTML page at the top of the window. The title to be used in the documentation generated by `oxdoc` can be set with the `'-windowtitle'` option. For example:

```
oxdoc *.ox -windowtitle "My documentation"
```

4.3 Adding icons

To make the generated HTML files look a bit nicer, `oxdoc` supports icons. This feature can be enabled by adding the `'-icons'` option to the command line, or by adding the following line to your `oxdoc.xml` configuration file.

```
<option name="enableicons" value="yes" />
```

This option will generate references to icons in the `icons` subdirectory of your project folder. The easiest way to get started with this is to copy the standard set of icons from the `icons/` subdirectory in your `oxdoc` installation directory into a new subdirectory called `icons` under your project folder.

In case you want to use a different set of icons: the standard set consists of a number PNG files by the names `xxx.png` and `xxx_s.png` where `xxx` is a base file name, e.g. `index.png` and `index_s.png`. The `'_s'` stands for `'small'`. The dimensions of large icons are 32 times 32 pixels, and the dimensions of the small icons are 16 times 16 pixels. Although these are the dimensions in the `icons/` subdirectory, you are in no way obliged to use those specific dimensions.

Chapter 5

Configuration

5.1 Location of configuration files

`oxdoc` is configured by means of the file `oxdoc.xml` and by command line parameters. `oxdoc` parses its parameters in the following order:

1. The `oxdoc.xml` file in the directory in which `oxdoc.jar` is located. For example, on Windows operating systems, this file may be found in `c:\program files\oxdoc\bin`.
2. The `oxdoc.xml` file in the current working directory.
3. The command-line parameters.

This means that any parameter that is specified at some stage in the order above overrides that setting specified in earlier stages. It is a good idea to put computer-specific settings in the `bin` directory and project-specific settings in project directories.

5.2 Lay-out of `oxdoc.xml`

A configuration file looks something like this:

```
<oxdoc>
<option name="latex"      value="c:\texmf\miktex\bin\latex.exe" />
<option name="dvipng"    value="c:\texmf\miktex\bin\dvipng.exe" />
<option name="tempdir"   value="c:\temp\" />
</oxdoc>
```

This file specifies values for three options. More option values can be added to this file as required. See [Overview of available settings](#).

5.3 Command line configuration

It is also possible to specify settings through command line arguments by adding `-parameter value` to the command line. For example,

```
oxdoc -latex c:\bin\latex.exe *.ox
```

specifies a value for the `latex` setting. The names of the command line parameters correspond exactly to the settings in `oxdoc.xml`. The `yes/no` options (`-icons` and `-showinternals`) are specified without parameter. Also, if the parameter contains spaces, one should put the value between double quotes. For example:

```
oxdoc -latex "c:\program files\miktex\bin\latex.exe" -formulas latex
      -icons -showinternals *.ox
```

5.3.1 \LaTeX settings

`oxdoc` uses \LaTeX in combination with `dvipng` to generate PNG (Portable Network Graphics) files from formulas within comments. In order to get this working, you'll need a working distribution of \LaTeX (e.g. MiKTeX if you're using Windows) and `dvipng` (which comes with MiKTeX). It is then important to set the paths to the `latex` and `dvipng` executables. It is recommended to do this in the `oxdoc.xml` file in the `bin` directory of your `oxdoc` installation.

At startup, `oxdoc` checks whether it can find the executables required for \LaTeX support. If it can't find one or more of these executables, it automatically turns off \LaTeX support. In that case, formulas are literally written in the output. Turning off \LaTeX support can also be done manually by setting the `formulas` setting to `plain` or to `mathml`.

It is also possible to specify extra \LaTeX packages to be included within formulas. This can be done by specifying the desired packages, separated by commas, in the option `latexpackages`. The specified packages will be included into every \LaTeX formula by means of the `\usepackage` command.

5.4 Overview of available settings

The following parameters can be set on the command line or in a configuration file.

5.4.1 Options for input

`include` Specifies include search path. On Windows, the paths should be separated by semicolons (;). On Linux, they should be separated by colons (:).

5.4.2 Options for output

`formulas` Chooses the way formulas are handled. Possible values: `latex`, `mathml`, `plain`. Default: `latex` if the required executables can be found, `plain` otherwise.

`icons` Specifies whether the generated HTML files should contain references to icons. Possible values: `yes`, `no`. If set to `yes`, then the generated HTML files will contain references to icons in the `icons/` folder. See Section 4.3 on how to implement these icons.

`outputdir` Specifies the directory in which `oxdoc` writes its output. Defaults to the current working directory.

<code>projectname</code>	Specifies the name of the project. This name will appear in the project home page.
<code>showinternals</code>	Specifies whether internal methods and fields (i.e. methods and fields that are private, protected, or marked with the <code>@internal</code> clause) are included in the documentation. Possible values: <code>yes</code> , <code>no</code> . See 3.6
<code>windowtitle</code>	Specifies the title that will appear in the window caption in your web browser.

5.4.3 Options for third-party utilities

<code>dvipng</code>	Specifies the full path of the executable <code>dvipng</code> . For MiKTeX users, this can be found under the <code>miktex\bin</code> subdirectory of the MiKTeX installation path.
<code>imagepath</code>	Specifies the directory in which <code>oxdoc</code> writes the images that represent \LaTeX formulas. Defaults to the subdirectory 'images' in the specified output directory.
<code>imagebgcolor</code>	The background color for LaTeX images. Possible values: either an HTML color in the form <code>#RRGGBB</code> , where <code>RR</code> , <code>GG</code> and <code>BB</code> are hexadecimal representations of the intensity of the red, green, and blue components, or <code>transparent</code> . Examples: <code>#0000FF</code> for bright blue. The use of this is to make sure that the background color of the \LaTeX generated formula images coincides with the background color of the HTML pages. The default, <code>transparent</code> , generates PNG files that have transparent backgrounds. Unfortunately, not every web browser supports such transparent backgrounds, hence this option.
<code>latex</code>	Specifies the full path of the \LaTeX compiler. For MiKTeX users, this can be found under the <code>miktex\bin</code> subdirectory of the MiKTeX installation path.
<code>latexpackages</code>	Specifies what \LaTeX packages should be loaded for inline \LaTeX formulas. These packages are loaded in \LaTeX files through the usual <code>\usepackage{...}</code> command. Multiple packages can be specified by separating them by commas.
<code>tempdir</code>	Specifies the directory that <code>oxdoc</code> can use for temporary files. Defaults to the current working directory.