User's Manual for `oxdoc`

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

January 30, 2010

oxdoc is a small software package 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 sourcecode 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.

Contents

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 using the Setup program

In order to install oxdoc, download setup-0.975alpha.exe file from the SourceForge website, run it, and follow the instructions. In order to use LaTeX formulas, make sure to specify the location of MiKTeX (or any other LaTeX distribution). The setup program automatically creates a program group in the start menu from which the oxdoc graphical user interface is available.

1.2.2 Manual installation on Windows

Follow the following steps to manually install oxdoc:

1. Unzip the package `oxdoc-xxx-bin.zip` into a suitable folder. For example, `c:\program files\oxdoc`.
2. Edit the file `oxdoc.bat` and alter the `oxdoc` variable in this file. This file looks as follows:

   ```
   @echo off
   set oxdoc=c:\program files\oxdoc
   java -jar "%oxdoc%\bin\oxdoc.jar" %1 %2 %3 %4 %5 %6 %7 %8 %9
   ```

   The second line in this file has to point to the directory in which the `oxdoc` files have been unzipped. The same holds for the file `oxdocgui.bat`.
3. Optionally, copy the files `oxdoc.bat` and `oxdocgui.bat` into a folder in the Windows search path, e.g. `c:\windows`. This way, it is possible to run `oxdoc` from any folder in the command prompt.
4. Edit `oxdoc.xml` in the `bin` directory. This file contains general settings for `oxdoc`. See the Configuration section for more information.

To test whether `oxdoc` works, run the batch file `oxdoc.bat` from the command line. It should display a short description of the program options.

1.2.3 Manual installation on Linux

Follow the following steps to manually install `oxdoc`:

1. Unzip the package `oxdoc-xxx-bin.tar.gz` into a suitable folder. For example, the `oxdoc` folder in your user directory.
2. Edit `oxdoc.xml` in the `bin` directory. This file contains general settings for `oxdoc`. See the Configuration section for more information.

To test whether `oxdoc` works, run the script file `oxdoc` from the `bin` directory. 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). If you installed oxdoc using the setup program in Windows, this interface can be accessed from the Start menu.

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:

```
/** The multivariate Normal distribution $\mathcal{N}(\mu, \Sigma)$.
An instance of a NormalDistribution class generates realizations of a random
variable $X$ with probability density function
$$f(x) = |\Sigma|^{-1/2}(2\pi)^{-n/2}
       \exp\left(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right).$$

@author Y. Zwols

@example To generate 20 samples from a standard normal distribution,
the following code can be used:
<pre>
decl Dist = new NormalDistribution(0, 1);
decl Z = Dist.Generate(20);
</pre>
**/
class NormalDistribution {
   /** This vector denotes the mean of this normal distribution object. **/
   decl m_vMu;

   /** This matrix denotes the variance/covariance matrix of this normal distribution object. **/
   decl m_mSigma;

   NormalDistribution(const vMu, const mSigma);
   virtual Generate(const cT);
   virtual Dim();
}

/** Create a new instance of the NormalDistribution class with parameters $\mu$
```

```
and $\Sigma$.
@param vMu The mean $\mu$ of the normal distribution
@param mSigma The variance/covariance matrix
@comments The dimension of the multivariate normal distribution is deduced
from the dimensions of the arguments. **/
NormalDistribution::NormalDistribution(const vMu, const mSigma) {
   expectMatrix("vMu", vMu, rows(vMu), 1);
   expectMatrix("mSigma", mSigma, rows(vMu), rows(vMu));
   m_vMu = vMu;
   m_mSigma = mSigma;
}

/** Generate a vector of realizations. The length of the sample is given
by the argument cT.
@param cT Number of samples
**/
NormalDistribution::Generate(const cT) {
   return rann(cT, Dim());
}

/** The dimension of the multivariate normal distribution.
@comments This is deduced from the arguments given to the constructor. **/
NormalDistribution::Dim() {
   return rows(m_vMu);
}
```

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 `&nbsp;` (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 inclusing 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.

```
/** Abstract distribution class
    @see NormalDistribution, UniformDistribution **/
class Distribution {
   ...
}
```

3.3 Using LaTeX-style formulas

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

```
<pre>
/** Calculate the OLS estimates for the model $y = X\beta$.
 @returns The OLS estimate $\hat\beta = (X'X)^{-1}X'y$.
 **/
regression(X, y) {
   return invert(X'X)*X'y;
}
</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’ flags is specified will be hidden from the documentation. For example:

```
/** Updates references. @internal **/
MyClass::updateReferences()
{
  ...
}
```

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. oxdoc looks for this file at two locations:

1. the directory in which oxdoc.jar is located (e.g. c:\program files\oxdoc\bin);
2. the current working directory.

Whenever available, settings are loaded from these files in that order. Parameters set in the current working directory configuration file override the general settings in the oxdoc folder.

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.

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 is 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 `enablelatex` setting to `no`.

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`.

5.4 Overview of available settings

The following parameters can be set in a configuration file:

`outputdir`	Specifies the directory in which `oxdoc` writes its output. Defaults to the current working directory.
`tempdir`	Specifies the directory that `oxdoc` can use for temporary files. Defaults to the current working directory.
`projectname`	Specifies the name of the project. This name will appear in the project home page.
`windowtitle`	Specifies the title that will appear in the window caption in your web browser.
`imagepath`	Specifies the directory in which `oxdoc` writes its images. Defaults to the specified output directory.
`dvipng`	Specifies the full path of the executable `dvipng`. For MiKTeX users, this can be found under the `miktex\bin` subdirectory of the MiKTeX installation path.
`latex`	Specifies the full path of the LaTeX compiler. For MiKTeX users, this can be found under the `miktex\bin` subdirectory of the MiKTeX installation path.
`enablelatex`	Turns on or off LaTeX support. Possible values: `yes`, `no`. Default: `yes` if the required executables can be found.
`latexpackages`	Specifies what LaTeX packages should be loaded for inline LaTeX formulas. These packages are loaded in LaTeX files through the usual `/usepackage...` command. Multiple packages can be specified by separating them by commas.