xPapers Administrator's Manual

Table of Contents

Version

This is version 1.0 of the manual for xPapers 1.0.

Basic installation

Architecture overview

xPapers is a web application written in Perl. It is built with the HTML::Mason template system, the Rose::DB::Object ORM, and FastCGI. Concretely, it consists in a FastCGI handler, a large number of Mason templates, and a number of custom Rose-derived classes. The Rose classes depend on a MySQL database with SphinxSE installed, and the whole set up more or less depends on the Apache2 web server. xPapers also comes with a larger number of standalone scripts. Setting all this is up is non-trivial, but this manual should take you through the process in a couple of hours.

System requirements

xPapers should run fine on any Linux-based system with Perl 5.8 or more, though all development and testing so far has been done on Debian derivatives, and using such a system would probably make things significantly easier for you (we recommend Ubuntu server). xPapers may also work in a Cygwin environment on Windows, but this has not been tested and would probably require some patching. You need root access to the system.

Hardware-wise, it all depends on the amount of traffic you have to support. At the time of writing, PhilPapers.org runs on an 8-core machine with 32GB of ram. We comfortably serve about 300K visitors per month with this hardware. As a rule of thumb, you should plan to have one resident fastcgi handler per concurrent non-static request you have to handle. Each resident process potentially consumes about 150MB of ram and should have about half a CPU. Try also to have enough ram to fit all your data so the MySQL database files are always fully cached. For example, the PhilPapers database currently has 10GB (for 350K papers and 20K users) and we want to support about 20 concurrent dynamic requests, so we should plan a minimum of 13GB just for xPapers. In addition, we need to have a lot of free RAM for some RAM-hungry maintenance processes like the automated classifier (at least 8GB). We'd be fine with 24GB, but we got 32 to be safe.

Instructions

With a bit of luck, you should be able to get xPapers going in little time by following these instructions to the letter on a brand new Debian or Ubuntu system. Installing on a RHEL-style system will require that you do a bit of research to find the RPM packages that correspond to the deb packages used here. These instructions assume that you don't have any of the external dependencies (e.g. MySQL, Apache) installed and will only use them for xPapers. Adjust as appropriate.

APT dependencies

```
sudo apt-get install automake libtool libncurses5-dev g++ sysstat aspell
libaspell-dev perl apache2 libapache2-mod-fastcgi git
```

Note: we need to install MySQL from source to compile it with Sphinx. If you have MySQL installed with your distribution's packaging system, you should preferably remove it.

xPapers source

You may already have done this, but if not check out the xPapers distrib where you would like xpapers to live. We recommend that you set it up in /home/xpapers because that's the default directory we use in all the examples provided. Using this directory is going to make things significantly simpler for you. Also, it makes sense if you set up a separate xpapers user to run your cron jobs and all, which we also recommend. Let's assume you want to do this:

```
sudo adduser xpapers
```

[answer the questions]

While at it, we recommend adding yourself (or xpapers) to the www-data group for convenience. This will allow you to run the cron scripts and restart the xpapers processes. Edit /etc/group as required.

Now let's pull the xpapers code.

```
cd /home

sudo git clone git@github.com:xPapers/xPapers.git
```

(remove /home/xpapers if required: sudo rm -rf /home/xpapers)

```
sudo mv xPapers xpapers

sudo chown -R DESIRED_USER xpapers
```

let's set an environment variable to the xPapers home directory for future reference:

```
export XPAPERS=/home/xpapers
```

So that you don't always have to use the -I argument on the Perl interpreted, it's recommended that you link all the sub-directories under $XPAPERS/lib under /etc/perl, or some other directory in your PERL5LIB environment variable (or you can add $XPAPERS/lib) to PERL5LIB.

```
cd /etc/perl

sudo ln -s $XPAPERS/lib/xPapers .

sudo ln -s $XPAPERS/lib/Rose .

sudo ln -s $XPAPERS/lib/AI .

sudo ln -s $XPAPERS/lib/Statistics .

sudo ln -s $XPAPERS/lib/Language .
```

IMPORTANT: The Rose and AI directories contain a few files that redefines classes in the Rose::* and AI::* CPAN package hierarchies—they are patched classes that fix some bugs in these packages that the maintainers haven't (to our knowledge) fixed yet. **Make sure that /etc/perl comes before the other locations where the CPAN packages live for the right files to be used by Perl.**

MySQL and Sphinx

Download MySQL 5.1, source distribution. These instructions **will not work with 5.5** and above, though xPapers and Sphinx should work with 5.5 and above.

Download Sphinx 1.10-beta or more

MySQL with SphinxSE

```
cp -r [Sphinx source dir]/mysqlse [Mysql source dir]/storage/sphinx

cd [MySQL source dir]

sh BUILD/autorun.sh

./configure  --prefix=/usr/local/mysql \
--exec-prefix=/usr/local/mysql \
--with-charset=utf8 \
```

```
--with-mysqld-user=mysql \
--localstatedir=/var/lib/mysql \
--enable-assembler \
--with-unix-socket-path=/var/run/mysqld/mysqld.sock \
--with-plugins=sphinx,innobase \
--sysconfdir=/usr/local/mysql/etc

make

sudo make install

sudo /etc/init.d/mysql restart
```

Then configure MySQL:

```
sudo groupadd mysql

sudo useradd -g mysql mysql

cd /usr/local/mysql

sudo chown -R mysql .

sudo chgrp -R mysql .

sudo bin/mysql_install_db --user=mysql

sudo chown -R root .

sudo chown -R mysql /var/lib/mysql

sudo cp share/mysql/my-medium.cnf /usr/local/mysql/etc/my.cnf

sudo cp share/mysql/mysql.server /etc/init.d/mysql

sudo update-rc.d mysql defaults 19

sudo ln -s /usr/local/mysql/bin/mysql /usr/bin/mysql

sudo ln -s /usr/local/mysql/bin/mysql_config /usr/bin/mysql_config



/usr/local/mysql/bin/mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u
root mysql #time zone data



sudo /etc/init.d/mysql start
```

You will want to customize the /usr/local/mysql/etc/my.cnf configuration file.

Now if you open the mysql client (sudo mysql) and do "show engines;" you should see the SPHINX engine. Consult the Sphinx documentation for more help.

While at it, create a MySQL user:

```
mysql>CREATE USER 'xpapers'@'localhost' IDENTIFIED BY 'some_pass';

mysql>GRANT ALL PRIVILEGES ON *.* TO 'xpapers'@'localhost';
```

Now compile and install Sphinx:

```
cd [Sphinx source dir]

./configure --with-mysql=/usr/local/mysql \
--prefix=/usr/local/sphinx \
--exec-prefix=/usr/local/sphinx

make

sudo make install
```

You need to set up Sphinx's configuration. Our configuration file should work just fine if your XPAPERS directory is /home/xpapers:

```
sudo cp $XPAPERS/etc/sphinx.conf /usr/local/sphinx/etc/
```

Then **change the database connection details in that file**.

Then build your first index:

```
sudo /usr/local/sphinx/bin/indexer --all
```

Perl modules

Make sure cpan works.

```
# cpan

cpan> o conf init
```

(choose to do everything automatically)

```
cd $XPAPERS

sudo perl -Ilib bin/setup/cpan.pl etc/dependencies.txt

sudo cpan S/SK/SKIMO/FCGI-0.67.tar.gz
```

```

```

We need to download a specific version of the FCGI package because newer versions are incompatible with HTML::Mason 1.0.

IMPORTANT: The Algorithm::SVM package has been broken for a long time and may require manual intervention to install. If the installation fails, follow the instructions found on RT here: https://rt.cpan.org/Public/Bug/Display.html?id=43669 Once you've done that, run the cpan.pl script again, it will pick up where it left.

The default answers to all questions should be fine.

Bibutils

A binary distribution of bibutils is in $XPAPERS/bin/bibutils by default. If you are not on 64 bit Linux you will need to recompile these programs from source. Just uncompress the tarball provided in $XPAPERS/src and follow the instructions inside. Then copy or link the resulting executables on top of those in $XPAPERS/bin/bibutils. Bibutils is required for some bibliographic file import/export options.

Setup and tweak a minimal xPapers config

First, decide how you'll call your site internally – I'll take 'mysite' as example. Next copy the example config files to /etc:

```
sudo cp -ar $XPAPERS/etc/xpapers.d /etc/
```

Edit this file, and replace 'philpapers' by 'mysite' throughout. Then create an empty directory with your site's name under $XPAPERS/sites:

```
mkdir $XPAPERS/sites/mysite
```

That's where you'll put all your site-specific configuration (aside for the config files in /etc).

Aside for changing 'philpapers' for 'mysite', you should also change **the database settings and the $SUBJECT variable** in /etc/xpapers.d/main.pl. The rest can be adjusted later.

Of course, there are more configuration variables available than this. For a complete list, check the documentation of the xPapers::Conf and xPapers::Conf::* packages. Each variable in these packages can be overriden by a variable under /etc/xpapers.d/. There is more doc inside these packages themselves for the more obscure options.

Initialize your database and xpapers subdirectories

```
cd $XPAPERS

perl -Ilib bin/setup/init-database.pl

sudo perl -Ilib bin/setup/mkdirs.pl

sudo perl -Ilib bin/setup/compileyui.pl

```

Apache configuration

Use $XPAPERS/etc/apache2 as an example set of Apache configuration files. The important (xPapers-specific) files are mason.conf, sites-enabled/default-site, and mods-enabled/fastcgi*. Adapt to your needs. You should change the admin email, domain name, etc, but this default config should work out of the box.

```
Sudo mv  /etc/apache2 /etc/apache2-backup

sudo cp -ar $XPAPERS/etc/apache2 /etc/apache2
```

You also need to set up the xpapers init script:

```
sudo cp $XPAPERS/bin/start /etc/init.d/xpapers

sudo update-rc.d xpapers defaults 21
```

Now this script will start apache2, so we need to remove the apache2 init script:

```
sudo update-rc.d -f apache2 remove
```

IMPORTANT: You should always control apache2 through /etc/init.d/xpapers or your memory cache may become corrupt.

The xpapers init script also controls Sphinx for convenience. It assumes the paths used here, so you may have to modify it if you are using other paths. Now try it:

```
sudo /etc/init.d/xpapers restart
```

You might see a warning about Sphinx not running, that's OK. Our init script is a little sloppy but it works. (A better init script would be a welcome contribution.)

Now try your new xPapers site with a web browser (http://localhost, or whatever you domain name is).

Sign up for external services

xPapers is configured to connect up with several external services:

- **reCAPTCHA (necessary)**: you need to get reCAPTCHA credentials and save them in /etc/xpapers.d/main.pl where indicated
- **Facebook (optional)**: unless you want to unplug the Facebook option in the Followers module, register your app and input your credentials in /etc/xpapers.d/main.pl
- **Amazon Affiliate (optional)**: unless you plan to turn off Amazon links, register yourself as an affiliate on the US, UK and CA sites (the only three supported for now) and register your credentials in /etc/xpapers.d/main.pl

Set up your cron tasks

A large number of tasks are accomplished through cron scripts, including periodic re-building of the Sphinx index for search. There is sample crontab in $XPAPERS/etc/contrab.example. Not all the services are necessary, but many are. The example includes all required tasks but Sphinx re-indexing. The latter is done as follows:

```
$SPHINX/bin/indexer --all --rotate
```

The example might also have a section marked as PhilPapers-specific that you should remove on pain of generating lots of errors.

Three administration tools

As administrator, you have three main ways of interacting with xPapers.

The administration console

The first thing you should do once you have xPapers up and running is to try to access the admin console. You must first log in using the admin user you've created when initializing the database. Once you've done this, you should see an 'Admin menu' link at the top-right of the screen. This will take you to the admin console, which contains a good number of self-explanatory options.

The console normally displays a histogram of recent activity. The histogram is going to look funny when you first sign in due to a lack of data.

Email alerts

The maintenance scripts you should run for your site will send you numerous email messages. Admin emails will always have a subject line that begins with 'Admin notice:'. You should pay attention to these emails, because things will go wrong sometimes. You should pay particular attention to emails that have the word 'failed' in the subject header. We use that as a convention to mark serious errors. Typically, something will have gone so wrong that a script will have panicked and aborted. (Yes, that's meant to be used in conjunction with keyword filters in your mail client.)

The pink zones

In addition to the tools found as part of the admin console, there are many important admin tools scattered around the web interface. By default, they appear on a pink background to mark them out clearly as admin-only features. You'll see them only when signed in with your admin account.

Managing content intake

xPapers has the capability to handle content input from a variety of sources. This section describes how this is set up and managed for each source type.

User submissions

Direct user submissions (including user uploads) should require no setup. However, user submissions of full bibliographies should be monitored, as they can swamp your site with unwanted content. There is a tool for this purpose in the administration's console. You can check submissions (imports) and reverse them.

Personal pages

The personal page tracking feature of xPapers rely on opp-tools, as separate piece of open source software (see xpapers.org for a current link to opp-tools). Opp-tools serves as a back-end for xPapers: once you have them set up properly, all your interactions with opp-tools will be through the xPapers web interface.

The interaction works as follows. First you set up opp-tools, leaving its database of pages to track empty. To modify the pages to track, use the corresponding features in the xPapers web interface. This updates a couple of database tables on the xPapers side. You then use the synchronize_opp.pl script provided with xPapers to sync with opp. To get results from opp-tools, you use its RSS output CGI script – you read it like any other RSS feed using xPapers' RSS ingestion module.

OAI archives

All the machinery required for tracking OAI archives is included with xPapers. You should first populate your database of archives using OpenDOAR. A script is provided to make this easy (some tweaking might be required to suit your requirements): $XPAPERS/bin/harvest/opendoar-fetch.pl. You should begin by harvesting all archives added by the script to see what the results are like, then tweak as appropriate. You'll probably want to drop some archives whose content is irrelevant to your site.

The content harvested by the OAI harvester is determined in good part by the regular expressions defined in the config file (look for the string 'OAI'). These define what OAI sets get selected, and what items are retained when not harvesting a whole set (when doing a 'partial' harvest).

The script that performs the actual harvesting is $XPAPERS/bin/harvest/oai.pl.

CrossRef's Metadata Services

CrossRef's metadata services (crossref.org) are the easiest way to get data for journal articles. There is a tool for setting this up in the admin console. You don't have to input your authentication credentials anywhere, it's IP based.

CrossRef is fairly low maintenance, but you should check periodically that it's working all right.

The script that performs the actual harvesting is $XPAPERS/bin/harvest/crossref.pl.

RSS feeds

xPapers supports RSS feeds with PRISM tags. PRISM is a new standard for detailed citation data in RSS feeds (See philpapers.org/help/feeds.html for details on the format). How this works is largely self-explanatory from the admin console, but a few points are worth noting:

- When registering a new feed, you can choose between a type of 'journal', 'archive', or 'other'. This determines what sort of sanity checks the harvester applies. In particular, if you say 'journal' and no PRISM tag for the publication name is found by the harvester, it's going to complain.
- You shouldn't touch any of the other settings aside from 'name' and 'url'.
- The name is arbitrary, it's just for you.
- A column of the feed listing shows you how many papers were found on each of the N last attempts. If you see something like '0,0,0,0,10,10,12', a feed has stopped turning up papers, which is a bad sign.

The script that performs the actual harvesting is $XPAPERS/bin/harvest/feeds.pl.

Z39.50

Z39.50 is for harvesting libraries catalogues. For example, the Library of Congress exposes its catalogue through Z39.50. By default, xPapers is setup to harvest from them. In fact, there are so many irregularities between implementations of Z39.50 that it will probably not work out of the box with any other catalogue. The basic settings are defined by $Z3950_* variables in the configuration files.

The selection of works from a catalogue is based in part on the Library of Congress call numbers. This is set up from the 'Z39.50' part of the admin console. There you tell xPapers how to handle certain call number ranges. Each LOC call number has the form 'AANNNN', where 'AA' is a string of one to two letters (the 'class'), and 'NNNN' is a (possibly decimal) number. Each entry in the admin console picks a range of numbers (the 'NNNN' part) within a specific class, and tells xPapers what to do with items matching that range. The behaviour is defined by telling xPapers how many keyword matches it needs to find in the item's metadata to accept it (typical values are 0, 1 and 2). The relevant keywords are defined by the @MARCXML_KEYWORDS configuration variable. It's also possible to specify additional keywords to use for a specific range.

The script that performs the actual harvesting is $XPAPERS/bin/harvest/z3950.pl.

The category system

The category system is an important part of the xPapers framework. This PhilPapers help page explains how it works in some details:

http://philpapers.org/help/categorization.html

The category system is intended to be maintained by appointed editors (PhilPapers has hundreds of editors). The roles and tools available to editors are described here:

http://philpapers.org/help/editors.html

Most of this material should apply to any site running xPapers. However, you only need to have area-level categories to have a functioning site.

Categories are modified using the “edit category structure” option in the admin menu.

IMPORTANT: after adding areas, you must run the $XPAPERS/bin/setup/mkforums.pl script to create the forums associated with area-level categories.

Automatic categorization facilities

xPapers comes with two mechanisms to help the categorization process:

1. Journal-based categorization. When you visit a journal's page (“browse journals”) as admin, you can assign it a category using the search box in the pink zone. If you do assign a category, all new articles from this journal will automatically be put in the category by the $XPAPERS/bin/routine/autocatj.pl script.
2. AI-based categorization. There is also machine-learning based categorizer. It works well for area-level categorization once you have enough items to train it. The training script generates self-test statistics that will indicate you the quality of the categorization that's possible. The stats are output in $XPAPERS/var/cat_data, where the state of the train categorizer is also saved. Both $XPAPERS/bin/routine/train-categorizer.pl and $XPAPERS/bin/routine/categorizer.pl take two numeric arguments representing the id of the category to limit categorization to (typically 1, for the whole category system) and the depth level of the categories to categorize into (typically 1, for area-level categorization). Different categorizers and statistics are saved in var/cat_data for each combination of parameters. The

categorize.pl script also accept the string 'recent' as third parameter to indicate that only recently added items should be categorized.

Customizing the look and feel of your site

There are two main ways to make easy changes to the look and feel of your site: configuration variables and image overrides.

Configuration variables

The first and most obvious way is by overriding configuration variables in your /etc/xpapers.d/ files. This allows you to change the HTML colour scheme and many other things. See the files themselves for documentation on the specific options.

Changing the image files

You'll definitely want to provide your own logo, and you'll probably want to change other image files as well. This is done not by changing the files included with xPapers in $XPAPERS/assets/raw (where the default images are located), but by defining overrides for them. You define an override by adding an imagine with a corresponding name to $XPAPERS/sites/mysite/raw. For example, to change the logo, save your logo as $XPAPERS/sites/mysite/raw/logo.gif.

Advanced customization

The override mechanism discussed above for images work for all files under $XPAPERS/assets:

- **assets/mason**: the Mason components which make up much of xPapers' UI (Mason components are HTML documentations that mix in some Perl a la PHP)
- **assets/conf**: the domain-specific knowledge files. These files tell xPapers a number of things, for example, how to rewrite certain journal names, or what URL are publisher URLs (for proxy browsing).
- **assets/raw**: the images as previously discussed.

Any file under $XPAPERS/assets/ can have a counterpart under your site-specific tree in $XPAPERS/sites/mysite. When a counterpart is found, it is used instead of the default file. This is the way to change the behaviour of xPapers. **You should never change any file under $XPAPERS/assets**.

Note: replacing a Mason component in this way is a lot like overriding a method in a class: if your new component calls a component XYZ and you don't have a site-specific counterpart of XYZ, the default component will get called.

Here are some particularly important components you will find yourself wanting to override before you go public (paths expressed relative to $XPAPERS/assets/mason):

- header.html: the header
- footer.html: the footer
- credits.html: the file creditings the authors of the site
- intro.html: the introductory blurb that appears on the home page

More information

For more information, please visit http://www.xpapers.org.