Twyne

Release 0.35.2

Sam Wilson

Feb 07, 2022

CONTENTS:

Welcome to the **user manual for Twyne**, an open-source web-based content management system that is based around the traditional blog format of chronological posts that can contain writing, photographs, and lots of metadata. Twyne aims to operate within the indieweb, a philosophy of website architecture that prioritises openness, interoperability, and a sense of personal control for people running their own websites.

This user manual is aimed at people running Twyne for their own websites. It covers installation, configuration, and general usage. Any questions about anything in here should be asked on the issue tracker on Github (to do this, you will need a Github user account, which is free). For any question that you ask, we will endevour to fix up the manual to make it clearer about that topic.

CHAPTER

ONE

INSTALLING AND UPGRADING

1.1 Prerequesites

To install and use Twyne, you need the following software on your web server:

1. a web server with PHP 7.3 or above;
2. MariaDB (10.3 or above) or MySQL (5.7 or above)
3. command-line access to that server;
4. the Git version control system;
5. the PHP package manager, Composer;
6. ExifTool for working with embedded photo metadata; and
7. ImageMagick for creating multiple sizes of images.

1.2 Downloading

First, clone the latest version of the source code into a non web-accessible location on your server:

```
git clone https://github.com/samwilson/twyne.git /var/www/twyne
```

Create a new database and grant access to it to a new user. Add the details of this database and the user's credentials to the `.env.local` file in the `DATABASE_URL` key:

```
DATABASE_URL=mysql://user_name:password@127.0.0.1:3306/db_name
```

Then install the application with the included script, which checks out the latest version and installs all dependencies with Composer:

```
/var/www/twyne/bin/deploy.sh prod /var/www/twyne
```

Next, set up your web server to serve the `public/` directory as the root of the new web site. For Apache, this could be something like the following:

```
<VirtualHost *:80>
    ServerName example.org
    RewriteRule ^/(.*)$ https://example.org/$1 [L,R=permanent,QSA]
</VirtualHost>
<VirtualHost *:80>
```

(continues on next page)

(continued from previous page)

```
    ServerName www.example.org
    RewriteRule ^/(.*)$ https://example.org/$1 [L,R=permanent,QSA]
</VirtualHost>
<VirtualHost *:443>
    ServerName example.org
    DocumentRoot /var/www/twyne/public/
    SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
    <Directory "/var/www/twyne/public/">
        RewriteRule ^index\.php$ - [L]
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule . /index.php [L]
    </Directory>
</VirtualHost>
```

Where `example.org` is replaced by your own domain name.

Your new Twyne site should now be live. Navigate to the home page, and continue the set-up process as detailed in the *Configuration* section of this manual.

1.3 Upgrading

Upgrading is as simple as re-running the `deploy.sh` script. This will checkout the latest version of the code, install dependencies, and update the database as required. This script can be run automatically on a daily or weekly basis (as a Cronjob or Scheduled Task, for instance).

Prior to Twyne 0.30.0, location data was not read from any uploaded files that had GPS data in them (i.e. usually photographs). To retrospecively add this data, run

```
./bin/console twyne:extract-gps
```

CHAPTER

TWO

CONFIGURATION

All the important parts of configuration of a Twyne site happen in the `.env.local` file. The minimum that should be set are as follows:

```
APP_SECRET=a-long-random-string
APP_MAIL_SENDER=admin@example.org
APP_LOG_RECIPIENT=admin@example.org
MAILER_DSN=smtp://username:password@smtp.gmail.com
```

Other configuration (that is likely to be changed occasionally during the normal operation of the site) happens on the *Settings* page, accessible via the main menu.

2.1 Uploaded files

There are two options for file storage: the local filesystem, and S3-compatible object stores (such as those from AWS, Digital Ocean, OVH, Dreamhost, etc.).

The local filesystem is the default, and files will be stored in the `var/app_data/` directory. This location can be changed via `.env.local`:

```
APP_FS_DATA_DIR=/path/to/your/data_dir/
```

To use S3-compatible storage, set the following environment variables:

```
APP_FS_DATA_STORE=aws
APP_FS_AWS_REGION=
APP_FS_AWS_ENDPOINT=
APP_FS_AWS_BUCKET=
APP_FS_AWS_KEY=
APP_FS_AWS_SECRET=
```

Uploaded files will have various derivative files created for them, such as thumbnails of images. These files will be stored in the `var/app_temp/` directory by default. This directory can be changed in `.env.local`:

```
APP_FS_TEMP_DIR=/path/to/your/tmp_dir/
```

The temporary directory contains only files that will be regenerated as required (hence the name 'temporary'; these are however long-lived files). There is no need to back up this directory.

2.2 Maps

Maps are shown for individual Posts (both when viewing and editing), and for all posts on the `/map` page. The tiles used for these maps can be configured via the following four environment variables:

```
APP_MAP_TILES_VIEW_URL="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
APP_MAP_TILES_VIEW_CONFIG="{\"label\": \"OSM\", \"attribution\": \"&copy; <a href=
↪'https://openstreetmap.org/copyright'>OpenStreetMap contributors</a>\", \"maxZoom\": \
↪"19\"}"
APP_MAP_TILES_EDIT_URL="https://server.arcgisonline.com/ArcGIS/rest/services/World_
↪Imagery/MapServer/tile/{z}/{y}/{x}"
APP_MAP_TILES_EDIT_CONFIG="{\"label\": \"Esri\", \"attribution\": \"&copy; Esri &mdash;␣
↪Source: Esri, i-cubed, USDA, USGS, AEX, GeoEye, Getmapping, Aerogrid, IGN, IGP, UPR-
↪EGP, and the GIS User Community\", \"maxZoom\": \"19\"}"
```

The values shown here are the defaults as given in `.env`, and they use the OpenStreetMap rendered map, and Esri satellite imagery.

The `*_CONFIG` variables are JSON, and must be well-formed and correctly escaped. To check that you have set their values correctly, browse to `/map-config.json` and confirm that you see the desired structured output.

Logged-in users can switch between the two sets of tiles via a layer selector in the top right of the map. The `label` value for each `*_CONFIG` variable is what sets the user-visible label that's shown in the selector.

The 'EDIT' tiles (by default, the satellite imagery) are only shown to logged-in users, because it can be useful to use a restricted or expensive source for these tiles.

The Leaflet Providers tool can be useful for finding compatible tiles and their required configuration (as well as things such as whether they require registration).

CHAPTER

THREE

SETTINGS

The site settings area of Tywne is where various aspects of a site can be configured and viewed. Only administrators can view and change settings. Settings are different from *Configuration*, which happens in the `.env.local` file. The difference between the two systems is that settings are likely to change during normal operation of a site, but configuration is not.

3.1 Site title

The site title is used in a few places, such as the main website header, and in emails that are sent by the site. It should not contain any HTML.

3.2 Allow user registrations?

This setting can be used to turn off new user registrations. By default, anyone can register a new account on the site (having an account doesn't grant any special permissions though).

3.3 API key

If you use the CLI client, you'll need to set an API key. This should be any random alphanumeric string, and must be kept secret.

3.4 Redirects

In a subpage of Settings is the redirects table. This is a list of URL redirects and responses. It's used by the software when posts are deleted or tags merged, to make sure that former URLs don't stop working.

It can also be used by site owners to add their own redirects, for example to maintain backwards compatibility of URLs from a different CMS that was previously used for the same domain, or as a URL shortening system for simplifying any external or internal URLs.

Redirects have three parts: a path, a destination, and a status. The path always starts with a slash and is the local URL path (without protocol or domain) that will be redirected. The destination is either a local path (i.e. starting with a slash) or an external fully-qualified URL (i.e. to redirect to a different site). The status is the HTTP status code that will be used. A special status is `410`, which is used when you do not want to redirect a URL, but rather tell the user that whatever was once at that URL is now *gone*. No destination is required for those redirects (which aren't really redirects!).

3.5 Custom CSS and Scripts

It is possible to add custom CSS and Javascript that is loaded on every page. This can be useful for things such as modifying the appearance of your site or adding statistics-logging Javascript. It is not a fully-fledged theming or plugin system, but can certainly be used for many common tweaks.

To add or edit the custom CSS or Javascript, navigate to the *Settings* page, and go to the *Styles* or *Scripts* tab. Each of these is a single text box, the contents of which will be loaded verbatim on almost every page on the site. The only pages on which it is not loaded are the two pages on which the code is edited, in order to avoid errors in the custom code making it harder to modify that same code.

There is no linting or error-checking of any sort done to the text that you enter, so be careful — this feature gives you the power to break the site! If for whatever reason you are not able to using the main navigation links, the two pages can be found at `https://example.com/settings/css` and `https://example.com/settings/js`.

Both code-editing text boxes have syntax-highlighting enabled.

CHAPTER

FOUR

POSTS

Posts are the core data structure of Twyne, forming a chronological series of writing and uploaded files. At their most basic, they have an author, a date and time, and a title or body. These and Posts' other attributes are explained in full below.

In addition to these attributes, Posts can also have *Tags* and *Syndications*.

4.1 ID

Each Post has an integer identifier, which forms part of its URL. For example, Post 123 has a URL of `https://example.org/P123`.

The `P` does not form part of the ID, but is added in many contexts to help distinguish Post IDs from other IDs such as those of Tags. Anywhere that a Post ID can be entered (for example, in the reply-to field), both the prefixed and un-prefixed forms are acceptable, and will be normalized to the prefixed form.

4.2 Date

Dates and times are stored in the UTC timezone, but can be entered in whatever timezone you prefer. They will be displayed in the user's timezone (when the site is viewed in a web browser; for other contexts such as RSS feeds, UTC is used).

4.3 Author

A Post must have an author (even if that author is the ever-prolific "anon."). A site's default author can be set with the `APP_MAIN_CONTACT` environment variable, which is a reference to the ID of whichever Contact should be the default (usually this is the first user, which is created at install-time).

4.4 Title

Titles are short unformatted strings. A Post does not have to have a title, in which case it will be displayed slightly differently in post lists and on its own page. Titles do not have to be unique.

4.5 Body

The body of a Post is the main, possibly multi-paragraph, text that is formatted according to Twyne's Markdown syntax.

4.6 Location

GPS coordinates can be saved for any Post. A link to the Wikimedia Geohack tool will be provided, as a map-pin emoji, for example: .

4.7 User group

A Post is only viewable a single User Group. By default this is the "Public" group.

Note that Users can be in multiple groups.

4.8 File

A Post can have an attached file, which can be an image (JPEG, PNG, or GIF format), or PDF.

When uploading a file such as a JPEG that has embedded GPS coordinates, the Post's Locaiton field will also be set (if it doesn't already have a location).

4.9 Original URL

If a Post is being syndicated from another site, it will have a link back to its canonical location on that site.

4.10 Replies

A Post can be in reply to another post. Often, that other post will be a syndication from another site. Replies in Twyne are similar to what are called 'comments' in other blogging systems (such as WordPress).

On any Post page there is a 'reply' link, which goes to a new-post form with the "In reply to" field already filled in.

CHAPTER

FIVE

TAGS

Tags are the primary way of grouping Posts together by topic; each Post can have any number of Tags. Each Tag has its own page (with a URL of the form `/Txx`, where `xx` is the Tag's ID) and on that page it has a title (which must be unique, and which can be changed whenever required), and optional description (which uses the same syntax as Post bodies).

A Tag's page has a list of Posts, which is paginated if there are more than ten posts. The first page has a URL of the form `/Txx` and subsequent pages `/Txx/page-n` (where `n` is 2 or greater). The layout of each post is the same as for the chronological listings of posts.

5.1 Wikidata

Tags can also be linked to Wikidata items. This is the way to show that a Tag's topic is *the same as* the concept represented by the Wikidata item. For example, a Tag with a title of "Douglas Adams" would have the Wikidata item ID of "Q42". Only one Tag can be linked to each Wikidata item. Adding a Wikidata link to a Tag adds a metadata table to the Tag's page, showing all the relevant metadata from the Wikidata item. If a row in the metadata table refers to another Tag that is also linked to Wikidata, then the value displayed will be a link to that Tag.

Wikidata items can be merged (when multiple are accidentally created for the same concept), but this doesn't affect the linkage from Twyne. The only change will be that the new ID for the item will be displayed in the metadata table; no data in the database is changed, because the old ID will remain on Wikidata as a permanent alias for the new one.

The metadata from Wikidata is fetched up every time someone views a Tag's page, and is cached for two weeks. The cache can be cleared (forcing it to fetch the latest data) by running the following CLI command:

```
./bin/console cache:pool:clear cache.app
```

5.2 Merging

Sometimes, multiple tags might be created that cover the same topic or concept. In this case, they can be merged. Merging tags moves all of a tag's posts to another tag, and then deletes the first tag (leaving behind a redirect, to ensure existing URLs do not break).

To merge a tag, first go to the tag that you want to merge and follow the 'Merge' link. Then, enter the ID of the tag into which you want to merge. The next step presents you with both tags and their details, as well as a form with which to modify the details of the destination tag (for example, to add to the existing description, or update the Wikidata ID).

Be warned that there is *no way* to undo a tag merge!

CHAPTER

SIX

SYNDICATIONS

Syndications are Twyne's implementation of the POSSE idea from the indieweb: to *Post on one's Own Site, and Syndicate Elsewhere*. This means that a Post is first created on the Twyne site, and once it's saved it's then posted to other sites and linked back to the canonical version on the Twyne site. The Syndication data is saved against the Post, as a URL and label pair. The label can be anything, and is often the name of the site on which the Syndication has been saved.

For example, a short one-sentence post with no title might be made in Twyne and then tweeted in Twitter. The tweet would have the content of the post and its URL. After the tweet has been made, and has been given its own URL, that URL (and the label 'Twitter') will be added as a Syndication on the original post in Twyne.

Because syndicating to other sites is so common, Twyne tries to make this as easy as possible. For instance, when first saving a Post it may be simplest to click 'Save and keep editing', in order to remain in the editing form while you switch over to the other site in a separate tab, and then come back to save the new Syndication URL and label. Or, for Wikimedia Commons, there is a special exporting system that makes it much more streamlined to upload a file to that repository (along with all its metadata, and a link back to the Twyne site). This is explained below.

Other quick methods of adding Syndications are planned. If you have any ideas, please open an issue on Github.

6.1 Copy to Wikimedia Commons

For syndicating files to Wikimedia Commons, you first need to become familiar with what Commons is, and what sorts of files are wanted on the project. It's best to make your first contributions via the Upload Wizard, rather than directly from Twyne, so you get an iea of how things generally work. Then, if you do want to set up the quick-upload-from-Twyne, you need to create a Bot Password. Enter your site's name in the "Bot name" field (in the "Create a new bot password" section) and grant it the following permissions:

- Edit existing pages
- Create, edit, and move pages
- Upload new files
- Upload, replace, and move files

After saving the new Bot Password, it will give you a username and password. These need to be added to your `.env.local` file:

```
APP_COMMONS_USERNAME=Your_Username@Bot_Name
APP_COMMONS_PASSWORD=TheBotPassword123ABC
```

Now, when editing a Post, a link to 'Copy to Wikimedia Commons' will appear below the syndications table. Clicking this will take you to a form that shows a preview of the file, and fields for all the required meatadata. The full wikitext of the page can be modified at this point, along with the Structured Data caption and the depicts statements. For more information about how to use each of these elements, please read the Commons documentation.