
wagtailtrans Documentation

Release 0.1.0

LUKKIEN

December 28, 2016

1	Table of contents	3
1.1	Getting started	3
1.2	Migrate your existing Wagtail site	5
1.3	Settings	8
1.4	Reference	9
1.5	Release notes	10

Wagtailtrans is a package that can be used to facilitate multi language sites in Wagtail. We developed wagtailtrans with two implementations in mind.

- A **synchronised** tree
- A **freeform** tree

Synchronised tree is a method used to keep the tree structure in all languages the same. Any changes in the tree of the default language will also automatically be done in the language tree(s).

See also:

The documentation about [*Synchronized translation trees*](#)

Freeform trees allows the customization of the language trees. Pages can still be copied with content into the preferred language but this is not automatic nor mandatory. moderators can decide how the page structure works for every language.

See also:

The documentation about [*Freeform translation trees*](#)

Table of contents

1.1 Getting started

To start using wagtailtrans in your project, take the following steps:

1.1.1 Installation

1. Install Wagtailtrans via pip

```
$ pip install wagtailtrans
```

2. Add wagtailtrans to your INSTALLED_APPS:

```
INSTALLED_APPS = [  
    # ...  
    'wagtailtrans',  
    # ...  
]
```

3. Add wagtailtrans.middleware.TranslationMiddleware to your MIDDLEWARE_CLASSES:

```
MIDDLEWARE_CLASSES = [  
    # ...  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'wagtailtrans.middleware.TranslationMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    # ...  
]
```

Note: Keep in mind wagtailtrans.middleware.TranslationMiddleware is a replacement for django.middleware.locale.LocaleMiddleware.

4. Optionally, add WAGTAILTRANS_TEMPLATE_DIR to your TEMPLATES[0]['DIRS']

Note: As of Wagtail 1.8 Page.get_admin_display_title is added which doesn't require overriding admin templates anymore, so if you're on Wagtail >= 1.8 you can skip this step.

```
from wagtailtrans import WAGTAILTRANS_TEMPLATE_DIR

TEMPLATES = [{
    # ...
    'DIRS': [
        WAGTAILTRANS_TEMPLATE_DIR,
    ],
    # ...
}]
```

1.1.2 Configuration

Before we start incorporating wagtailtrans in your project, you'll need to configure wagtailtrans for the behavior that best suits the need of your project. The required settings to consider here are:

- WAGTAILTRANS_SYNC_TREE
- WAGTAILTRANS_LANGUAGES_PER_SITE

Both settings are mandatory but provided with a default value, so if you want *synchronized* trees and no languages per site, you're good to go from here.

See also:

Complete reference about available settings: [Settings](#)

1.1.3 Incorporating

To start using wagtailtrans we first need to create a translation home page. This page will route the requests to the homepage in the right language. We can create a translation site root page by creating the `wagtailtrans.models.TranslatableSiteRootPage` as the first page under the root page.

In this example we will also make a `HomePage` which will be translatable. This is done by implementing the `wagtailtrans.models.TranslatablePage` next to Wagtail's `Page`

```
from wagtail.wagtailcore.models import Page
from wagtailtrans.models import TranslatablePage

class HomePage(TranslatablePage, Page):
    body = RichTextField(blank=True, default="")
    image = models.ForeignKey('wagtailimages.Image', null=True, blank=True, on_delete=models.SET_NULL)

    content_panels = Page.content_panels + [
        FieldPanel('body'),
        ImageChooserPanel('image')
    ]

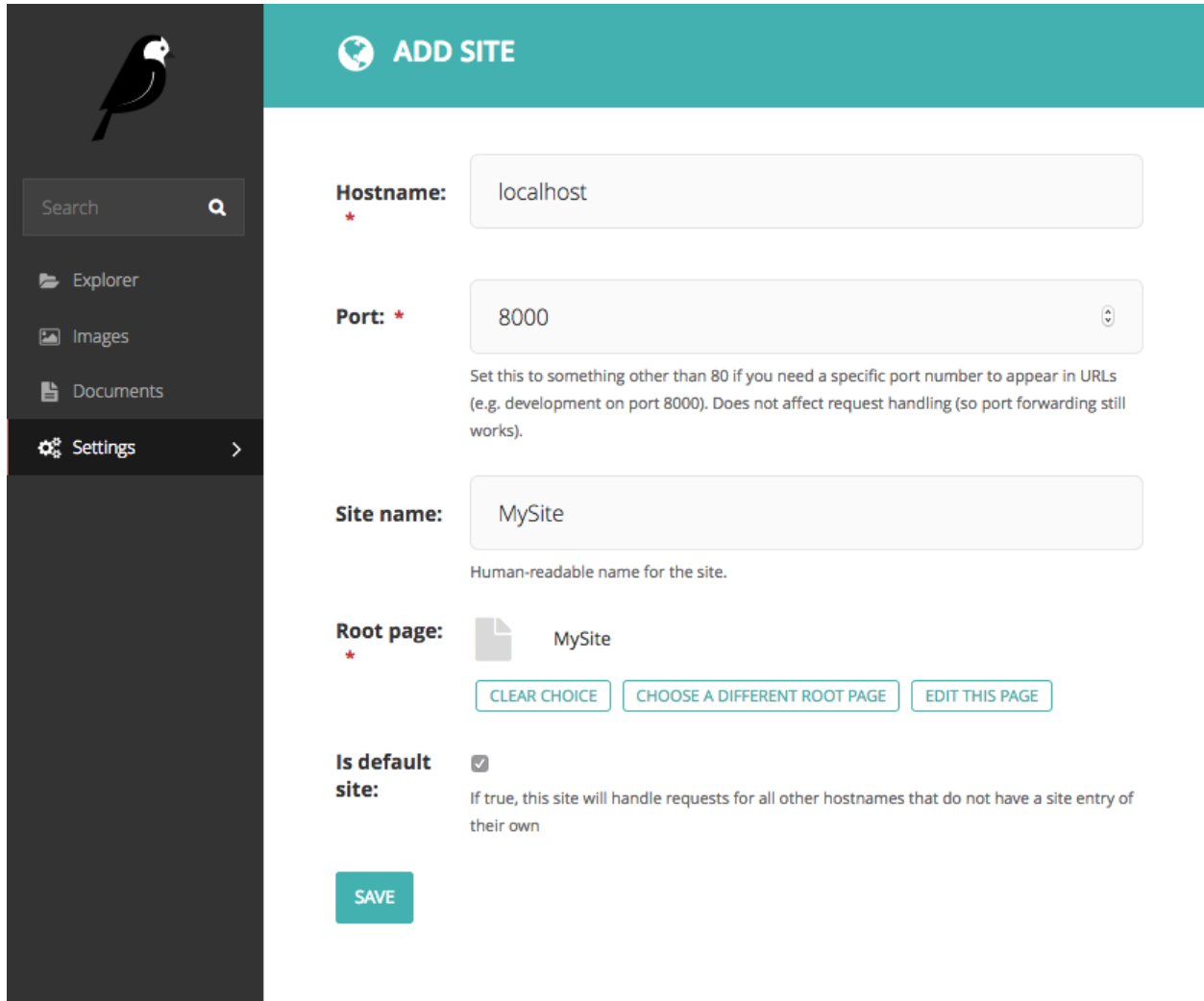
    subpage_types = [
        # Your subpage types.
    ]
```

This will create our first translatable page. To start using it we first need to migrate our database

```
$ python manage.py makemigrations
$ python manage.py migrate
```

Now run the server and under the page Root create a TranslatableSiteRootPage (MySite).

Next we need to create a site and point its `root_page` to our TranslatableSiteRootPage (MySite).



ADD SITE

Hostname:

Port: Set this to something other than 80 if you need a specific port number to appear in URLs (e.g. development on port 8000). Does not affect request handling (so port forwarding still works).

Site name: Human-readable name for the site.

Root page: MySite

Is default site: ☒ If true, this site will handle requests for all other hostnames that do not have a site entry of their own

SAVE

We now have the basics for a Translatable Site.

1.2 Migrate your existing Wagtail site

Migrating of already existing Wagtail content to Wagtailtrans can be quite difficult. Since there is no way to do this automatically, we've put some examples below to point you in the right direction.

Danger: Examples below contain custom database migrations, make sure you've created a back-up of your database before you start this migration process.

1.2.1 Non Wagtailtrans site

1. Install wagtailtrans

```
$ pip install wagtailtrans
```

2. Add wagtailtrans to INSTALLED_APPS

```
INSTALLED_APPS = [  
    # ...  
    'wagtailtrans',  
    # ...  
]
```

3. Add `wagtailtrans.models.TranslatablePage` to the existing `wagtail.wagtailcore.models.Page` models.

```
from wagtail.wagtailcore.models import Page  
from wagtailtrans.models import TranslatablePage  
  
class HomePage(TranslatablePage, Page):  
    # ...
```

4. Create a database migration file, when makemigrations is asking for a one-off default value for `translatablepage_ptr` you can fill in a fake value, since we're going to change this later.

```
$ python manage.py makemigrations <appname>
```

5. Update migrations file to add the newly `translatablepage_ptr_id` field in the required table.

Note: We've made some assumptions when creating this migration file, for example a default language `en`. Please make sure you've checked all the database queries and adjusted them according to your own setup before executing.

```
class Migration(migrations.Migration):  
  
    dependencies = [  
        ('wagtailtrans', '0006_auto_20161212_2020'),  
        ('pages', '0002_auto_20160930_1042'),  
    ]  
  
    operations = [  
        migrations.RunSQL(  
            """  
            BEGIN;  
  
            -- Remove constraints so we can edit our table  
            ALTER TABLE pages_homepage DROP CONSTRAINT pages_homepage_pkey CASCADE;  
  
            -- Add ``translatablepage_ptr_id`` field and copy the ``page_ptr_id`` content  
            ALTER TABLE pages_homepage ADD COLUMN translatablepage_ptr_id INTEGER UNIQUE;  
            UPDATE pages_homepage SET translatablepage_ptr_id=page_ptr_id;  
  
            -- Insert the required values in ``wagtailtrans`` table  
            INSERT INTO wagtailtrans_language (code, is_default, position, live) SELECT 'en', 't', 0,  
            INSERT INTO wagtailtrans_translatablepage (translatable_page_ptr_id, canonical_page_id, .  
  
            -- Add required indexes and constraints  
            ALTER TABLE pages_homepage ADD CONSTRAINT pages_homepage_translatablepage_ptr_id_e5b77cf  
            ALTER TABLE pages_homepage ALTER COLUMN translatablepage_ptr_id SET NOT NULL;  
            ALTER TABLE pages_homepage ADD PRIMARY KEY (translatablepage_ptr_id);
```

```

        COMMIT;
        """
        state_operations=[
            migrations.AddField(
                model_name='homepage',
                name='translatablepage_ptr',
                field=models.OneToOneField(auto_created=True, on_delete=django.db.models.deletion.CASCADE, to='pages.Page', preserve_default=False),
            ),
            migrations.AlterField(
                model_name='homepage',
                name='page_ptr',
                field=models.OneToOneField(auto_created=True, on_delete=django.db.models.deletion.CASCADE, to='pages.Page'),
            ),
        ],
    ],
)
]

```

1.2.2 Pre 0.1 Wagtailtrans site

Before the 0.1 final release we've made a backwards incompatible change by defining a custom `parent_link`, this is done to ease the process of migrate an existing Wagtail site to Wagtailtrans.

Migrating can be done by following these steps:

1. Update code where necessary, models inheriting from `wagtailtrans.models.TranslatablePage` should also inherit from `wagtail.wagtailcore.models.Page`

```

from wagtail.wagtailcore.models import Page
from wagtailtrans.models import TranslatablePage

class HomePage(TranslatablePage, Page):
    # ....

```

2. Create a database migration file, when `makemigrations` is asking for a one-off default value for `page_ptr` you can fill in a fake value, since we're going to change this later.

```
$ python manage.py makemigrations <appname>
```

3. Alter the migration file to add the `page_ptr_id` field to the database, update it with the right values, create the required indexes and constraints and update the ORM state with a separate state operation.

Note: We've made some assumptions when creating this migration file. Please make sure you've checked all the database queries and adjusted them according your own setup before executing.

```

class Migration(migrations.Migration):

    dependencies = [
        ('wagtailcore', '0029_unicode_slugfield_dj19'),
        ('pages', '0002_auto_20160930_1042'),
        ('wagtailtrans', '0006_auto_20161212_2020'),
    ]

    operations = [

```

```
migrations.RunSQL(
    """
    BEGIN;

    -- Add the ``page_ptr_id`` field in the DB.
    ALTER TABLE pages_homepage ADD COLUMN page_ptr_id INTEGER UNIQUE;
    UPDATE pages_homepage SET page_ptr_id=translatablepage_ptr_id;
    ALTER TABLE pages_homepage ALTER COLUMN page_ptr_id DROP DEFAULT;
    ALTER TABLE pages_homepage ALTER COLUMN page_ptr_id SET NOT NULL;
    ALTER TABLE pages_homepage ADD CONSTRAINT pages_homepage_page_ptr_id_5b805d74_fk_wagtail

    COMMIT;
    """,
    state_operations=[
        migrations.AddField(
            model_name='homepage',
            name='page_ptr',
            field=models.OneToOneField(auto_created=True, on_delete=django.db.models.deletion
            preserve_default=False,
        ),
    ],
),
]
```

1.3 Settings

There are a few settings which can be used to configure wagtailtrans to suit your needs, these settings need to be configured in your django settings module. All wagtailtrans settings are prefixed with WAGTAILTRANS_ to avoid conflicts with other packages used.

1.3.1 WAGTAILTRANS_SYNC_TREE

Default True

If set to False wagtailtrans will work with Freeform trees.

See also:

The documentation about *Synchronized translation trees*

See also:

The documentation about *Freeform translation trees*

1.3.2 WAGTAILTRANS_LANGUAGES_PER_SITE

Default False

If set to True wagtailtrans will allow you to define a default language and additional languages per site. This is mostly used in a multi site setup and allows you to define the languages per site, this way they can differ for all available sites.

1.4 Reference

1.4.1 Synchronized translation trees

Before you can start using synchronized trees, please be sure you followed all steps in: [Getting started](#).

If you specified the `WAGTAILTRANS_SYNC_TREE` in your settings as `True` you will be using the synchronized trees. This means that every change in your ‘canonical’ tree will also be done in the translated trees. To start using this we first need to create a default language (canonical). In your wagtail admin page in settings, select languages.

Fig. 1.1: We will be using English as our default language. Please do Note that only *ONE* default language can be used

Add any other language you will be using in your site but don’t check *is default*. for demonstration purposes we add German and Spanish.

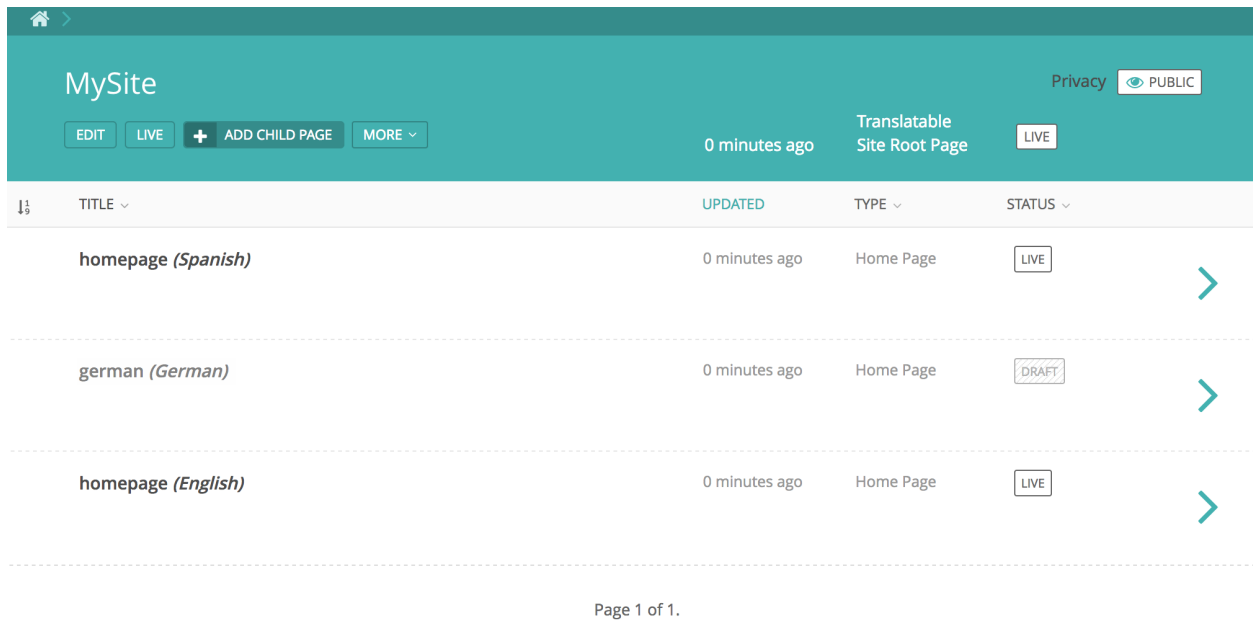
Now we go to our site root page (MySite) and create a new page. called *Homepage* after saving the Homepage copies of this page will be saved for each language. This will happen for every *TranslatablePage* instance created when `WAGTAILTRANS_SYNC_TREE = True`. This way language trees will always be synchronized.

With the creation of a language a new translator user group is created. This group will gain edit and publish permissions on each page with corresponding language.

Now any change made in the canonical tree will also be made in the translation trees.

1.4.2 Freeform translation trees

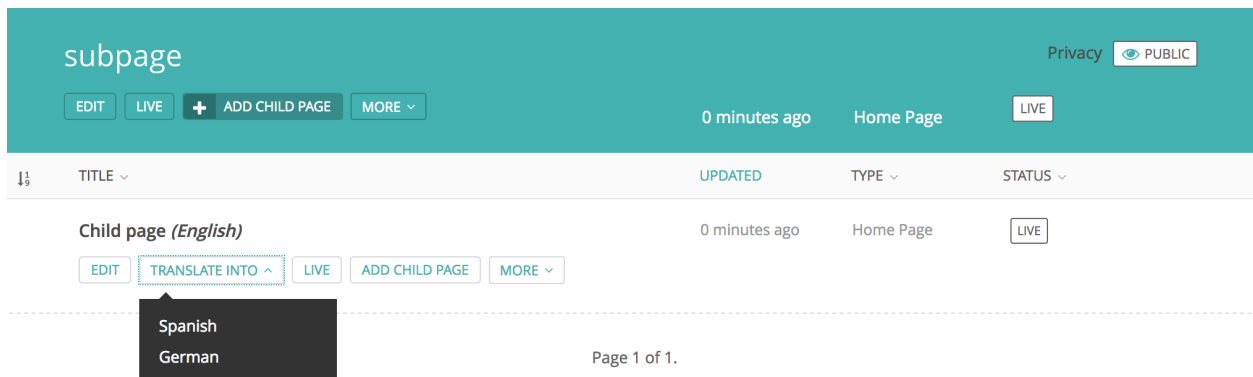
Before you can start using freeform trees, please be sure you followed all the steps in: [Getting started](#).



TITLE	UPDATED	TYPE	STATUS
homepage (Spanish)	0 minutes ago	Home Page	LIVE
german (German)	0 minutes ago	Home Page	DRAFT
homepage (English)	0 minutes ago	Home Page	LIVE

Page 1 of 1.

If you specified the `WAGTAILTRANS_SYNC_TREE` in your settings as `False` you will be using the freeform trees. the freeform trees are used if you do not want to keep your language trees the same for all languages. This enables the *translate into* button in the Wagtail admin. This will show all the languages that the page can be translated into.



TITLE	UPDATED	TYPE	STATUS
Child page (English)	0 minutes ago	Home Page	LIVE

Page 1 of 1.

- Once a language is selected you will be asked if you want to copy the content and where to move the page to.
- With this setting your page tree structure is free for all languages.
- The translator roles and groups will still be created as in the synchronized tree section.

1.5 Release notes

1.5.1 Wagtailtrans 0.1 release notes

- *What is new*
- *Backwards incompatible changes*

What is new

Since this is the first final release, all features are new.

Features

- Implement models following [Wagtail RFC9](#) by Tim Heap
- Force language of child pages to language of parent
- Support storing of translated pages
- Support copying content of canonical pages when creating translations
- Add translation information to the `TranslatablePage.settings_panels`
- Add dropdown page menu for adding translations
- Add Language admin-UI in settings-menu
- Add `WAGTAILTRANS_SYNC_TREE` setting to control which way trees behave
- Add `WAGTAILTRANS_TEMPLATE_DIR` to override the admin template dir (pre Wagtail 1.8)
- Add `WAGTAILTRANS_LANGUAGES_PER_SITE` setting to allow different page languages per site
- Add `SiteLanguages` as `SiteSetting` in settings-menu (`WAGTAILTRANS_LANGUAGES_PER_SITE`)
- Add `wagtailtrans.models.TranslatablePage.get_admin_display_title` to display the page language in the admin explorer (Wagtail 1.8+)

Backwards incompatible changes

In the progress of creating this release, we found out that some earlier design choices needed to be reworked in order to have a good maintainable package.

`include_self`

Before `wagtailtrans 0.1.0b4` the method `TranslatablePage.get_translations` implemented a kwarg to include itself in the `QuerySet` of translated pages. Since this isn't used within the `wagtailtrans` codebase we removed it because keeping it there made the codebase better maintainable.

`wagtailtrans.models.TranslatablePage.translatable_page_ptr`

As of `wagtailtrans 0.1` we've refactored the `TranslatablePage` model to have a custom `parent_link`, this way it's easier to migrate any existing Wagtail site for use with `wagtailtrans`. This is done by inheriting from `TranslatablePage` and `Page` for implementations of pages.

As for migrating, this is sadly not possible without some custom migrations, a way to do this is documented here: [Migrate your existing Wagtail site](#)