django-wiki Documentation

Release 0.3.dev20170627091715

Benjamin Bach

Jun 27, 2017

Contents

CHAPTER 1

Installation

Pre-requisite: Pillow

For image processing, django-wiki uses the Pillow library (a fork of PIL). The preferred method should be to get a system-wide, pre-compiled version of Pillow, for instance by getting the binaries from your Linux distribution repos.

Debian/Ubuntu

You need to get development libraries which Pip needs for compiling:

```
sudo apt-get install libjpeg8 libjpeg-dev libpng12-0 libpng12-dev
```

After that, install with `sudo pip install Pillow`. You might as well install Pillow system-wide, because there are little version-specific dependencies in Django applications when it comes to Pillow, and having multiple installations of the very same package is a bad practice in this case.

Mac OS X 10.5+

Ethan Tira-Thompson has created ports for OS X and made them available as a .dmg installer. Download and install the universal combo package here.

Once you have the packages installed, you can proceed to the pip installation. PIL will automatically pick up these libraries and compile them for django use.

Installing

To install the latest stable release:

```
pip install wiki
```

Install directly from Github (in case you have no worries about deploying our master branch directly):

```
pip install git+git://github.com/django-wiki/django-wiki.git
```

Upgrading

Always read the *Release notes* for instructions on upgrading.

Configuration

Configure `settings.INSTALLED_APPS`

The following applications should be listed - NB! it's important to maintain the order due to database relational constraints:

```
'django.contrib.sites',
'django.contrib.humanize',
'django_nyt',
'mptt',
'sekizai',
'sorl.thumbnail',
'wiki',
'wiki.plugins.attachments',
'wiki.plugins.notifications',
'wiki.plugins.images',
'wiki.plugins.macros',
```

Database

To sync and create tables, do:

```
python manage.py migrate
```

Configure `TEMPLATE_CONTEXT_PROCESSORS`

Add `'sekizai.context_processors.sekizai'` and `'django.core.context_processors.debug'` to `settings.TEMPLATE_CONTEXT_PROCESSORS`. Please refer to the Django settings docs to see the current default setting for this variable.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        # ...
        'OPTIONS': {
            'context_processors': [
                'django.contrib.auth.context_processors.auth',
                'django.template.context_processors.debug',
                'django.template.context_processors.i18n',
                'django.template.context_processors.media',
                'django.template.context_processors.request',
```

```
                'django.template.context_processors.static',
                'django.template.context_processors.tz',
                'django.contrib.messages.context_processors.messages',
                "sekizai.context_processors.sekizai",
            ],
        },
    },
]
```

Set `SITE_ID`

If you're working with fresh Django installation, you need to set the SITE_ID

```
SITE_ID = 1
```

User account handling

There is a limited account handling included to allow users to sign up. Its settings are shown below with their default values. To switch off account handling entirely, set `WIKI_ACCOUNT_HANDLING = False`.

```
WIKI_ACCOUNT_HANDLING = True
WIKI_ACCOUNT_SIGNUP_ALLOWED = True
```

After a user is logged in, they will be redirected to the value of `LOGIN_REDIRECT_URL`, which you can configure in your project settings to point to the root article:

```
from django.core.urlresolvers import reverse_lazy
LOGIN_REDIRECT_URL = reverse_lazy('wiki:get', kwargs={'path': ''})
```

Include urlpatterns

To integrate the wiki to your existing application, you should add the following lines at the end of your project's `urls.py`.

```
from wiki.urls import get_pattern as get_wiki_pattern
from django_nyt.urls import get_pattern as get_nyt_pattern
urlpatterns += [
    url(r'^notifications/', get_nyt_pattern()),
    url(r'', get_wiki_pattern())
]
```

The above line puts the wiki in */* so it's important to put it at the end of your urlconf. You can also put it in */wiki* by putting `'^wiki/'` as the pattern.

Note: If you are running `manage.py runserver`, you need to have static files and media files from `STATIC_ROOT` and `MEDIA_ROOT` served by the development server. `STATIC_ROOT` is automatically served, but you have to add `MEDIA_ROOT` manually:

```
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Please refer to the Django docs.

CHAPTER 2

Release notes

Release plan

The next release series **0.3** will support Django 1.11. Likewise, it will be the last series with Python 2 support. Series 0.3 is in development in the current master branch.

django-wiki 0.3 (unreleased)

- Test refactor: Use django-functest and separate WebTest from Selenium (Luke Plant) #634
- Django 1.11 compatibility (Luke Plant) #634
- Repo refactor: Moved `wiki` package to `src/` folder and test code to `tests/` #631
- New bootstrapped image insert dialog (Frank Loemker) #628
- Allow the HTML tag `<hr>` (Frank Loemker) #629
- Global History overview of page revisions (Frank Loemker and Maximilien Cuony) #627
- Move article support with redirects (Frank Loemker) #640
- Crop paginator window when there are >9 pages in a list (Frank Loemker) #646

django-wiki 0.2.4

- Hot-fix because of missing woff2 files #625

django-wiki 0.2.3

- Pulled Transifex translations and pushed source translations.

- Fix support for Py2 unicode in code blocks (Benjamin Bach) #607
- Support for Github style fenced codeblocks (Benjamin Bach) #618
- Cached articles showing up in wrong language (Benjamin Bach) #592
- Upgraded Bootstrap from 3.3.1 to 3.3.7 (Benjamin Bach) #620
- Upgraded bundled jQuery to 1.12.4 (Benjamin Bach) #620
- Setting `WIKI_MARKDOWN_HTML_STYLES` for allowing `style='..'` in user code (Benjamin Bach) #603
- Strip Markdown code in search result snippets (Benjamin Bach) #42

django-wiki 0.2.2

- Remove `wiki.decorators.json_view`, fixes server errors when resolving 404 links #604
- Replace usage of `render_to_response()` with `render()` #606
- Fix memory leak #609 and #611 (obtroston)
- Scroll bars and display area fixed for code blocks #601 and #608 (Branko Majic)
- Option `WIKI_MARKDOWN_SANITIZE_HTML` skips Bleach (warning: Don't use for untrusted code) #610 (Michal Hozza)
- Allow the HTML tag `<br>`. #613 (Frank Loemker)
- Add thumbnail size directive (example: `[image:123 size:large]`). #612 (Frank Loemker and @inflrscns)
- Fix error with absolute paths in wiki links (example: `[Sub-root](wiki:/sub-root)`) #616 (Benoit C. Sirois)
- Require `Django<1.11` #616 (Benoit C. Sirois)

django-wiki 0.2.1

- Lowercase slugs when creating new pages with `[[Like This]]` #595 (Eric Clack)
- Fix issues related to Bleach before Markdown processing esp. pertaining > characters. #596
- Remove `wiki.plugins.mediawikiimport` #597
- Pretty up the highligted code's line enumeration #598
- Customize codehilite in order to wrap highlighted code with scrollbars #598

django-wiki 0.2

- Translation updates from Transifex
  - Danish translation from 39% to 100% (Bo Holm-Rasmussen)
  - Updated languages since 0.1: Chinese, French, German, German, Russian, Spanish
- Added Django 1.10 support #563
- Security: Do not depend on markdown `safe_mode`, instead use `bleach`.

- Fix duplicate search results when logged in #582 (duvholt)
- Do not allow slugs only consisting of numbers #558
- Copy in urlify.js and fix auto-population of slug field in Django 1.9+ #554
- Fix memory leak in markdown extensions setting #564
- Updated translations - Languages > 90% completed: Chinese (China), Portuguese (Brazil), Korean (Korea), French, Slovak, Spanish, Dutch, German, Russian, Finnish.
- Taiwanese Chinese added (39% completed)
- Cleanup documentation structure #575

HTML contents

Bleach is now used to sanitize HTML before invoking Markdown.

HTML escaping is done before Markdown parsing happens. In future Markdown versions, HTML escaping is no longer done, and `safe_mode` is removed. We have already removed `safe_mode` from the default `WIKI_MARKDOWN_KWARGS` setting, however if you have configured this yourself, you are advised to remove `safe_mode`.

Allowed tags are from Bleach's default settings: `a`, `abbr`, `acronym`, `b`, `blockquote`, `code`, `em`, `i`, `li`, `ol`, `strong`, `ul`.

Please use new setting `WIKI_MARKDOWN_HTML_WHITELIST` and set a list of allowed tags to customize behavior.

Python and Django support

Support has been removed for:

- Python 2.6
- Django < 1.8
- South

django-wiki 0.1.2

- Remove unwanted items from default menu when `WIKI_ACCOUNT_HANDLING = False`. #545
- Fix broken soft-deletion and restoring of images, and "set revision" functionality #533
- Added responsiveness to tables by use of Bootstrap table-responsive class #552

django-wiki 0.1.1

- Several languages updated from Transifex
  - Slovak added **Thanks M Hozza**
  - Portuguese also added, but as copy of PT-BR (make changes as desired in Transifex)
- Brand new Account Settings page (email / password) **Thanks inflrscns**
- Testproject turned into Django 1.9 layout

- Replace context-processor dependent use of `{{ STATIC_URL }}` with `{% static %}`
- Bugfix for `pip install wiki` in an empty (no Django installed) virtualenv
- Precommit hooks added in repository
- Import statements sorted and codebase re-pep8'thed
- Log in page is now called "Log in" in `<title>` tag - **Thanks Eugene Obukhov**

django-wiki 0.1

Warning: If you are upgrading from a previous release, please ensure that you pass through the 0.0.24 release because it contains the final migrations necessary before entering the django-wiki 0.1+ migration tree.

If you are using django 1.7+ and have an old installation of django-wiki (which should be impossible since it wouldn't run) please downgrade to 1.6 as follows:

```
$ pip install wiki\<0.1 --upgrade  # Latest 0.0.24 release
$ pip install django\<1.7  # Downgrade django if necessary
$ python manage.py migrate  # Run 0.0.24 migrations
$ pip install wiki\<0.2 --upgrade  # Upgrade to latest 0.1 series
$ python manage.py migrate --delete-ghost-migrations  # Run migrations again,
                                                      # removing the (ghost)
                                                      # migrations from previous
                                                      # release

$ # Feel free to upgrade Django again
```

Supported

- Python 2.7, 3.3, 3.4, 3.5 (3.2 is not supported)
- Django 1.5, 1.6, 1.7, 1.8, 1.9
- Django < 1.7 still needs South, and migration trees are kept until next major release.

Breaking changes

wiki.VERSION as tuple

We want to follow Django's way of enumerating versions. If you want the old string version, use `wiki.__version__`.

Plugin API

Since Django 1.8 has started making warnings about *patterns* being deprecated, we've decided to stop using them by default. Thus, as with the future Django 2.0, we will use lists of *url* objects to store the urlconf of plugins. All the bundled plugins have been updated to reflect the change.

Django-mptt

We now depend on django-mptt 0.7.2+ for Django 1.8 compatibility.

django-wiki 0.0.24

This release is a transitional release for anyone still using an older version of django-wiki. The code base has been heavily refactored and this is hopefully the final release.

> **Warning:** 0.0.24 is mainly a transitional release, but new features and bug fixes are included, too.

Compatibility

- Django 1.5, 1.6 (That means Django 1.7 is **not** yet fully supported)
- South 1.0+ (if you are on an older South, you **need** to upgrade)
- Python 2.6, 2.7, 3.3, 3.4

Upgrading

Firstly, upgrade django-wiki through familiar steps with pip

```
$ pip install wiki --upgrade
```

During the upgrade, notice that django-nyt is installed. This replaces the previously bundled django_notify and you need to make a few changes in your settings and urls.

In `settings.INSTALLED_APPS`, replace *"django_notify"* with *"django_nyt"*. Then open up your project's urlconf and make sure you have something that looks like the following:

```
from wiki.urls import get_pattern as get_wiki_pattern
from django_nyt.urls import get_pattern as get_nyt_pattern
urlpatterns += patterns('',
    (r'^notifications/', get_nyt_pattern()),
    (r'', get_wiki_pattern())
)
```

Notice that we are importing *from django_nyt.urls* and no longer *django_notify* and that the function is renamed to *get_nyt_pattern*.

After making these changes, you should run migrations.

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

Notifications fixed

In past history, django-wiki has shipped with a very weird migration. It caused for the notifications plugin's table of article subscriptions to be removed. This is fixed in the new migrations and the table should be safely restored in case it was missing.

However, you may want to bootstrap subscription notifications in case you have run into this failed migration. You can ensure that all owners and editors of articles receive notifications using the following management command:

    python manage.py wiki_notifications_create_defaults

Troubleshooting

If you have been running from the git master branch, you may experience problems and need to re-run the migrations entirely.

```
python manage.py migrate notifications zero --delete-ghost-migrations
python manage.py migrate notifications
```

If you get *DatabaseError: no such table: notifications_articlesubscription*, you have been running django-wiki version with differently named tables. Don't worry, just fake the backwards migration:

```
python manage.py migrate notifications zero --fake
```

If you get `relation "notifications_articlesubscription" already exists` you may need to do a manual `DROP TABLE notifications_articlesubscription;` using your DB shell (after backing up this data).

After this, you can recreate your notifications with the former section's instructions.

News archive

April 15, 2017

0.2.3 released: Release notes

0.2.2 released: Release notes

February 27, 2017

0.2.1 released: Release notes

December 27, 2016

0.2 final released: Release notes

June 19, 2016

0.1.2 released: Release notes

May 6, 2016

0.1.1 released: Release notes

January 25, 2016

0.1 final released

December 26th, 2015

A new release 0.0.24.4 is out and has fixes for the Django `loaddata` management command such that you can create dumps and restore the dump. Notice, though, that `loaddata` only works for Django 1.7+.

Django 1.9 support is available in the current master, please help get a 0.1 released by giving feed back in the last remaining issues:

https://github.com/django-wiki/django-wiki/milestones/0.1

November 16th, 2015

Django 1.8 support is very ready and 0.1 is right on the doorstep now.

January 26th, 2015

After too long, the new release is out.

The wait was mainly due to all the confusing changes by adding support of Python 3 and readying the migrations for Django 1.7. But there's actually new features, too.

- Bootstrap 3.3.1 and Font Awesome 4 (Christian Duvholt)
- `django_nyt` instead of builtin `django_notify` (Benjamin Bach, Maximilien Cuony)
- `tox` for testing (Luke Plant)
- Appropriate use of gettext_lazy (Jaakko Luttinen)
- Fixed support of custom username fields (Jan De Bleser)
- Several fixes to the attachment plugin (Christian Duvholt)
- Errors on notifications settings tab (Benjamin Richter)
- Chinese translations (Ronald Bai)
- Finish translations (Jaakko Luttinen)
- Compatibility with custom user model in article settings (Andy Fang)
- Fixed bug when `[attachment:XX]` present multiple times on same line (Maximilien Cuony)
- Simple mediawiki import management command (Maximilien Cuony)
- Python 3 and Django 1.6 compatibility (Russell-Jones, Antonin Lenfant, Luke Plant, Lubimov Igor, Benjamin Bach)
- (and more, forgiveness asked if anyone feels left out)

CHAPTER 3

Plugins

Add/remove the following to your `settings.INSTALLED_APPS` to enable/disable the core plugins:

- `'wiki.plugins.attachments'`
- `'wiki.plugins.images'`
- `'wiki.plugins.notifications'`
- `'wiki.plugins.globalhistory'`

The notifications plugin is mandatory for an out-of-the-box installation. You can safely remove it from `INSTALLED_APPS` if you also override the **wiki/base.html** template.

CHAPTER 4

Customization

See *Settings* for the settings that can be used to configure django-wiki. Other ways to customize django-wiki for your use are listed below.

Templates

django-wiki can be customized by providing your own templates.

All templates used by django-wiki inherit from `wiki/base.html`, which in turn simply inherits from `wiki/base_site.html` (adding nothing). `wiki/base_site.html` provides a complete HTML page, but provides a number of blocks that you might want to override. The most useful are:

- `wiki_site_title`
- `wiki_header_branding`
- `wiki_header_navlinks`

These can be overridden to provide your own branding and links in the top bar of the page, as well as in browser window title. The `wiki/base_site.html` template uses Bootstrap 3, so the following example shows how to use this in practice, assuming you want a single link to your home page, and one to the wiki. Add the following as `wiki/base.html` somewhere in your `TEMPLATE_DIRS`:

```
{% extends "wiki/base_site.html" %}

{% block wiki_site_title %} - Wiki{% endblock %}

{% block wiki_header_branding %}
<a class="navbar-brand" href="/">Your brand</a>
{% endblock %}

{% block wiki_header_navlinks %}
<ul class="nav navbar-nav">
  <li class="active"><a href="{% url 'wiki:root' %}">Wiki</a></li>
```

```
</ul>
{% endblock %}
```

CHAPTER 5

Settings

The following settings are available for configuration through your project. All settings are customized by prefixing `WIKI_`, so for instance `URL_CASE_SENSITIVE` should be configured as `WIKI_URL_CASE_SENSITIVE`.

`wiki.conf.settings.`**`ACCOUNT_HANDLING`** **= True**
Sign up, login and logout views should be accessible

`wiki.conf.settings.`**`ACCOUNT_SIGNUP_ALLOWED`** **= True**
Signup allowed? If it's not allowed, logged in superusers can still access the signup page to create new users.

`wiki.conf.settings.`**`ANONYMOUS`** **= True**
Treat anonymous (non logged in) users as the "other" user group

`wiki.conf.settings.`**`ANONYMOUS_CREATE`** **= False**
Globally enable create access for anonymous users Defaults to ANONYMOUS_WRITE.

`wiki.conf.settings.`**`ANONYMOUS_UPLOAD`** **= False**
Default setting to allow anonymous users upload access (used in plugins.attachments and plugins.images).

`wiki.conf.settings.`**`ANONYMOUS_WRITE`** **= True**
Globally enable write access for anonymous users, if true anonymous users will be treated as the others_write boolean field on models.Article.

`wiki.conf.settings.`**`CACHE_TIMEOUT`** **= 600**
Seconds of timeout before renewing article cache. Articles are automatically renewed whenever an edit occurs but article content may be generated from other objects that are changed.

`wiki.conf.settings.`**`CAN_ADMIN`** **= None**
A function returning True/False if a user has permission to create new groups and users for the wiki.

`wiki.conf.settings.`**`CAN_ASSIGN`** **= None**
A function returning True/False if a user has permission to assign permissions on an article Relevance: changing owner and group membership

`wiki.conf.settings.`**`CAN_ASSIGN_OWNER`** **= None**
A function returning True/False if the owner of an article has permission to change the group to a user's own groups Relevance: changing group membership

wiki.conf.settings.**CAN_CHANGE_PERMISSIONS = None**
A function returning True/False if a user has permission to change read/write access for groups and others

wiki.conf.settings.**CAN_DELETE = None**
Specifies if a user has access to soft deletion of articles

wiki.conf.settings.**CAN_MODERATE = None**
A function returning True/False if a user has permission to change moderate, ie. lock articles and permanently delete content.

wiki.conf.settings.**CAN_READ = None**
A function returning True/False if a user has permission to read contents of an article + plugins Relevance: viewing articles and plugins

wiki.conf.settings.**CAN_WRITE = None**
A function returning True/False if a user has permission to change contents, ie add new revisions to an article Often, plugins also use this Relevance: editing articles, changing revisions, editing plugins

wiki.conf.settings.**CHECK_SLUG_URL_AVAILABLE = True**
When True, this blocks new slugs that resolve to non-wiki views, stopping users creating articles that conflict with overlapping URLs from other apps.

wiki.conf.settings.**EDITOR = u'wiki.editors.markitup.MarkItUp'**
The editor class to use – maybe a 3rd party or your own...? You can always extend the built-in editor and customize it!

wiki.conf.settings.**GROUP_MODEL = u'auth.Group'**
Choose the Group model to use. Defaults to django's auth.Group

wiki.conf.settings.**LOG_IPS_ANONYMOUS = True**
Do we want to log IPs of anonymous users?

wiki.conf.settings.**LOG_IPS_USERS = False**
Do we want to log IPs of logged in users?

wiki.conf.settings.**LOST_AND_FOUND_SLUG = u'lost-and-found'**
This slug is used in URLPath if an article has been deleted. The children of the URLPath of that article are moved to lost and found. They keep their permissions and all their content.

wiki.conf.settings.**MARKDOWN_HTML_ATTRIBUTES = {u'em': [u'class', u'id'], u'pre': [u'class', u'id'], u'code': [u'clas**
Dictionary of allowed attributes in Markdown article contents.

wiki.conf.settings.**MARKDOWN_HTML_STYLES = []**
Allowed inline styles in Markdown article contents, default is no styles (empty list)

wiki.conf.settings.**MARKDOWN_HTML_WHITELIST = [u'a', u'abbr', u'acronym', u'b', u'blockquote', u'code', u'em', u'**
List of allowed tags in Markdown article contents.

wiki.conf.settings.**MARKDOWN_KWARGS = {u'extension_configs': {u'toc': {u'title': <django.utils.functional.__proxy__ o**
Arguments for the Markdown instance, for instance a list of extensions to use. See: https://pythonhosted.org/Markdown/extensions/index.html

To set a custom title for TOC's:

```
WIKI_MARKDOWN_KWARGS = {'extension_configs': {'toc': _('Contents of this article
↪')}}
```

wiki.conf.settings.**MARKDOWN_SANITIZE_HTML = True**
Whether to use Bleach or not. It's not recommended to turn this off unless you know what you're doing and you don't want to use the other options.

wiki.conf.settings.**REVISIONS_MINUTES_LOOKBACK = 2**
Number of minutes for looking up REVISIONS_PER_MINUTES and REVISIONS_PER_MINUTES_ANONYMOUS

wiki.conf.settings.**REVISIONS_PER_HOUR = 60**
Maximum allowed revisions per hour for any given user or IP

wiki.conf.settings.**REVISIONS_PER_HOUR_ANONYMOUS = 10**
Maximum allowed revisions per hour for any given user or IP

wiki.conf.settings.**REVISIONS_PER_MINUTES = 5**
Maximum allowed revisions per minute for any given user or IP

wiki.conf.settings.**REVISIONS_PER_MINUTES_ANONYMOUS = 2**
Maximum allowed revisions per hour for any given user or IP

wiki.conf.settings.**SEARCH_VIEW = u'wiki.views.article.SearchView'**
Search view - dotted path denoting where the search view Class is located

wiki.conf.settings.**SHOW_MAX_CHILDREN = 20**
Maximum amount of children to display in a menu before going "+more" NEVER set this to 0 as it will wrongly inform the user that there are no children and for instance that an article can be safely deleted.

wiki.conf.settings.**STORAGE_BACKEND = <django.core.files.storage.DefaultStorage object>**
Django Storage backend to use for images, attachments etc.

wiki.conf.settings.**URL_CASE_SENSITIVE = False**
Should urls be case sensitive?

wiki.conf.settings.**URL_CONFIG_CLASS = u'wiki.urls.WikiURLPatterns'**
dottedname of class used to construct urlpatterns for wiki.

Default is wiki.urls.WikiURLPatterns. To customize urls or view handlers, you can derive from this.

wiki.conf.settings.**USE_BOOTSTRAP_SELECT_WIDGET = True**
User Bootstrap's select widget. Switch off if you're not using Bootstrap!

wiki.conf.settings.**USE_SENDFILE = False**
Use Sendfile

CHAPTER 6

Developer guide

Setting up a development environment

- Fork and clone the django-wiki repo from Github, `cd` into it.
- Create and activate a virtualenv for developing django-wiki. Ensure you are using recent setuptools and pip.
- Install the requirements:

```
$ pip install --upgrade pip setuptools
$ pip install -e .
$ pip install pytest pytest-django pytest-pythonpath pytest-cov mock django-
↪functest
```

Running the test project

In order to quickly get setup, there is a project in the git repository.

The folder **testproject/** contains a pre-configured django project and an sqlite database. Login for django admin is `admin:admin`. This project should always be maintained, but please do not commit changes to the SQLite database as we only care about its contents in case data models are changed.

Tests

Running tests

To run django-wiki's tests, run `make test` or `./runtests.py`

If you want to test for more **environments**, install "tox" (`pip install tox`) and then just run `tox` to run the test suite on multiple environments.

To run **specific tests**, see `./runtests.py --help`.

To include Selenium tests, you need to install chromedriver and run `./runtests.py --include-selenium`. For tox, do:

```
INCLUDE_SELENIUM_TESTS=1 tox
```

Writing tests

Tests generally fall into a few categories:

- Testing at the model level. These test cases should inherit from `tests.base.TestBase`.
- Tests for views that return HTML. We normally use django-functest for these, especially if the page involves forms and handling of POST data. Test cases should inherit from `tests.base.WebTestBase` and `tests.base.SeleniumBase` - see `tests.core.test_views.RootArticleViewTestsBase`, `RootArticleViewTestsWebTest` and `RootArticleViewTestsSelenium` for an example.

  (In the past the Django test Client was used for these, and currently there are still a lot of tests written in this style. These should be gradually phased out where possible, because the test Client does a poor job of replicating what browsers and people actually do.
- Tests for views that return JSON or other non-HTML. These test cases should inherit from `tests.base.DjangoClientTestBase`.

There are also other mixins in `tests.base` that provide commonly used fixtures for tests e.g. a root article.

Warning: Views should be written so that as far as possible they work without Javascript, and can be tested using the fast WebTest method, rather than relying on the slow and fragile Selenium method. Selenium tests are not run by default.

Types of Contributions

Report Bugs

Report bugs at https://github.com/django-wiki/django-wiki/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

Django-wiki could always use more documentation, whether as part of the official django-wiki docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/django-wiki/django-wiki/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *django-wiki* for local development.

1. Fork the *django-wiki* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-wiki.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-wiki
$ cd django-wiki/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ tox -e lint
$ py.test  # Run tests in current environment
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/django-wiki/django-wiki/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests.test_django-wiki
```

Roadmap

The best way to contribute is to use our Github issue list to look at current wishes. The list is found here:

https://github.com/django-wiki/django-wiki/issues/

If you want to add a feature, consider writing a plugin. Please create an issue to discuss whether your plugin idea is a core plugin (`wiki.plugins.*`) or external plugin. If there are additions needed to the plugin API, we can discuss that as well! A discussion is always welcome in a Github issue.

Generally speaking, we need more **unit tests** to improve coverage, and new features will not be accepted without tests. To add more stuff to the project without tests wouldn't be fair to the project or your hard work. We use coverage metrics to see that each new contribution does not significantly impact test coverage.

CHAPTER 7

Tips & FAQ

FAQ

Q: Why can't I move articles?

A: Moving articles is not trivial. Here are a couple of reasons:

- Other articles may link to them.
- Permissions may change if you move the articles into a different hierarchy
- We keep revisions of stuff, so the action of moving an article will create a new revision.
- ...but what if the revision is reverted and we had automatically renamed stuff?

Because it isn't trivial to move articles, the work has delayed somewhat.

Resources:

- Pull Request #461
- Issue #154

Q: Why do I keep getting *"This slug conflicts with an existing URL."*

A: When validating a slug, django-wiki will verify through *Settings*.‘‘WIKI_CHECK_SLUG_URL_AVAILABLE‘‘ (default: `True`) that the URL is not already occupied.

So if you keep getting an error that the "slug" isn't available, it's probably because you left another URL pattern interfearing with django-wiki's by letting your pattern (regexp) be too open. Forgetting a closing `$` is a common mistake.

Disqus comment embed

This page describes how to embed the Disqus comment system on each wiki page.

Put the following as `wiki/base.html` somewhere in your `TEMPLATE_DIRS`:

```
{% extends "wiki/base_site.html" %}
{% load sekizai_tags %}

{% block wiki_body %}
  {{ block.super }}
  {% block wiki_footer_logo %}
  {% endblock wiki_footer_logo %}
  {% if selected_tab == 'view' %}
    {% addtoblock "js" %}
    <script type="text/javascript">
    (function(){
      $("#wiki-footer p").eq(0).after('<div id="disqus_thread"></div>')
    })();
    var disqus_shortname = 'your_disqus_shortname';
    (function() {
      var dsq = document.createElement('script'); dsq.type = 'text/javascript'; dsq.
↪async = true;
      dsq.src = '//' + disqus_shortname + '.disqus.com/embed.js';
      (document.getElementsByTagName('head')[0] || document.getElementsByTagName('body
↪')[0]).appendChild(dsq);
    })();
    </script>
    {% endaddtoblock %}
  {% endif %}
{% endblock wiki_body %}
```

Replace `your_disqus_sortname` to your disqus sortname.

See also in *Customization*.

Quick tips

1. **Account handling:** There are simple views that handle login, logout and signup. They are on by default. Make sure to set `settings.LOGIN_URL` to point to your login page as many wiki views may redirect to a login page.
2. **Syntax highlighting:** Python-Markdown has a pre-shipped codehilite extension which works perfectly, so add something like:

```
WIKI_MARKDOWN_KWARGS = {
    'extensions': [
        'footnotes',
        'attr_list',
        'headerid',
        'extra',
        'codehilite',
    ]
}
```

to your settings. Currently, django-wiki ships with a stylesheet that already has the syntax highlighting CSS rules built-in. Oh, and you need to ensure `pip install pygments` because Pygments is what the codehilite extension is using!

3. **Project Templates:** Create new django-wiki projects quickly and easily using django-wiki project templates https://github.com/django-wiki/django-wiki-project-template

CHAPTER 8

django-wiki

Django support

The below table explains which Django versions are supported.

Release	Django	Upgrade from
0.3 (unreleased)	1.8, 1.9, 1.10, 1.11	0.2
0.2	1.8, 1.9, 1.10	0.1
0.1	1.5, 1.6, 1.7	0.0.24
0.0.24	1.4, 1.5, 1.6 1.7 (unstable)	0.0.?

For upgrade instructions, please refer to the Release Notes

Translations (Transifex)

Django-wiki has almost fully translated into 7 languages, apart from the default (English). But please help out in adding more languages! It's very easy, you don't even need to be a programmer.

https://www.transifex.com/django-wiki/django-wiki/

Demo

A demo running the latest `master` is available here, sign up for an account to see the notification system.

https://demo.django.wiki

Community

Please use our IRC or mailing list (google group) for getting in touch on development and support. Please do not email developers asking for personal support.

- #django-wiki on irc.freenode.net
- django-wiki@googlegroups.com
- twitter:djangowiki

THIS IS A WORK IN PROGRE...

Currently, the model API is subject to smaller changes, and the plugin API seems pretty stable.

South is used so no database changes will cause data loss. In order to customize the wiki, best idea is to override templates and create your own template tags. Do not make your own hard copy of this repository in order to fiddle with internal parts of the wiki – this strategy will lead you to lose out on future updates with highly improved features and plugins. Possibly security updates as well!

The release cycle has already begun, so you can administer django-wiki through Pypi and pip.

All views are class-based, however don't take it as an encouragement to extend them, unless you are prepared to modify both templates and view classes every time there is an update.

Contributing

Please read our Developer Guide

Manifesto

Django needs a mature wiki system appealing to all kinds of needs, both big and small:

- **Be pluggable and light-weight.** Don't integrate optional features in the core.
- **Be open.** Make an extension API that allows the ecology of the wiki to grow in a structured way. Wikipedia consists of over 1100 extension projects written for MediaWiki. We should learn from this.
- **Be smart.** This is the map of tables in MediaWiki - we'll understand the choices of other wiki projects and make our own. After-all, this is a Django project.
- **Be simple.** The source code should *almost* explain itself.
- **Be structured.** Markdown is a simple syntax for readability. Features should be implemented either through easy coding patterns in the content field, but rather stored in a structured way (in the database) and managed through a friendly interface. This gives control back to the website developer, and makes knowledge more usable. Just ask: Why has Wikipedia never changed? Answer: Because it's knowledge is stored in a complicated way, thus it becomes very static.

Docs

See the docs/ folder, or read them at:

http://django-wiki.readthedocs.io/en/latest/

If you wish to add something, please ask in the google group or raise an issue if you're in doubt about whether something might change.

Background

Django-wiki is a rewrite of django-simplewiki, a project from 2009 that aimed to be a base system for a wiki. It proposed that the user should customize the wiki by overwriting templates, but soon learned that the only customization that really took place was that people forked the entire project. We don't want that for django-wiki, we want it to be modular and extendable.

As of now, Django has existed for too long without a proper wiki application. The dream of django-wiki is to become a contestant alongside Mediawiki, so that Django developers can stick to the Django platform even when facing tough challenges such as implementing a wiki.

Q&A

- **Why is the module named just `wiki` ?** Because when we tried `pip install wiki`, it returned "No distributions at all found for wiki", so we had to make up for that!
- **What markup language will you use?** Markdown. The markup renderer is not a pluggable part but has been internalized into core parts. Discussion should go here: https://github.com/django-wiki/django-wiki/issues/76
- **Why not use django-reversion?** It's a great project, but if the wiki has to grow ambitious, someone will have to optimize its behavior, and using a third-party application for something as crucial as the revision system is a no-go in this regard.
- **Any support for multiple wikis?** Yes, in an sense you can just imagine that you always have multiple wikis, because you always have hierarchies and full control of their permissions. See this discussion: https://github.com/django-wiki/django-wiki/issues/63

Requirements

Please refer to current release to see exact version dependencies. And make note that Pillow needs to have certain build dependencies satisfied on your host system.

- Django
- Markdown
- django-mptt
- django-sekizai
- sorl-thumbnail
- Pillow (Python Imaging Library)
- Python>=2.7 or Python>=3.2

Acknowledgements

- The people at edX & MIT for finding and supporting the project both financially and with ideas.
- django-cms for venturing where no django app has gone before in terms of well-planned features and high standards. It's a very big inspiration.
- django-mptt, a wonderful utility for inexpensively using tree structures in Django with a relational database backend.
- spookylukey, jluttine, duvholt, valberg, jdcaballerov, yekibud, bridger, TomLottermann, crazyzubr, and everyone else involved!

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

W

Index

W