# Lithium

***Release 1.0.1***

**September 27, 2017**

# Contents

CHAPTER 1

# Lithium

Lithium is a set of applications written for the Django web framework. Lithium's goal is to provide every generic application a user may want on their website, such as a blog, forum, wiki, etc. Lithium could be installed to work as a content management system, my own personal website is ran by Lithium, and a few django apps such as flatpages.

## First steps

- *Download*
- *Installation*
- *Contributing*

## Applications

Applications included with the latest version of Lithium:

- *Blog*
- *Forum*
- *Contact*
- *Wiki*

## Applications which are not finished

These applications are still being developed, wheather they are testing, or not even started yet. This list is more of a list of what's left to-do:

- Store

- Profiles
- Media

## Getting Help

If you have any problems with Lithium, or have any suggestions you can contact kylef, or ask in the IRC Channel: #lithium at irc.freenode.org.

## Other stuff

- *Base template (base.html)*

CHAPTER 2

# Quick install guide

First you will need a working Django installation, then you can follow one of the many steps below to install lithium.

Once you have installed Lithium, look at the documentation for any *Lithium applications* on how to add it to your Django project. You must also create a *base template (base.html)*.

**Note:** Lithium does not require django.contrib.admin, but it is recommended that it is installed, most of the Lithium apps can be controlled via django.contrib.admin

## Directly from the git repository

This is the preferred way to install Lithium, but it does require the git source code repository system to be installed on your computer, although this is the preferred way, if you do not already have git. It will be easier to install via the other options. Some of the commands used in this are only available on UNIX-alike systems, it may not work on windows.

It is very easy to install lithium via git, first you need to open a terminal. Then type the following after first using `cd` command to move into the directory you wish to install lithium in. I use the location: `/home/kylef/git/` to install lithium in.

```
$ git clone git://github.com/kylef/lithium.git
$ ln -s lithium/lithium <PYTHONPATH>
```

## Using a package-management tool

This is one of the easiest ways to install Lithium. There are two different package-management tools which you could use:

### pip

pip is one of the more popular package-management systems for python. You can find documentation, and how to install pip itself here. Once you have pip installed and running, simply type:

```
$ pip install lithium
```

### easy_install

Another option is to use easy_install, first you need to install easy_install. You can find documentation and how to install easy_install here. Once you have easy_install up and running, just type:

```
$ easy_install lithium
```

## Manually installing

To manually install lithium, you will first need to download lithium. Once you have a copy of lithium, open it and run:

```
$ python setup.py install
```

This will install lithium and will require administrative privileges on your computer as it is a system-wide install.

CHAPTER 3

# How to get Lithium

Lithium is available under the BSD license. Lithium requires the latest development Django, or Django 1.1 once it is finished. Lithium will run on any system that Django will run on. Lithium requires a patch for Django that I wrote: http://code.djangoproject.com/ticket/10285

## Get the latest development version

It is recommended that you use the latest development version, we have not reached version 1.0 yet. Lithium 1.0 is quite far away, until then, you should use the development release.

### Via tar.gz / zip archive

You can download the latest version in a archive created directly from the git repository via the following links:

- tar.gz: http://github.com/kylef/lithium/tarball/master
- zip: http://github.com/kylef/lithium/zipball/master

### Via git repository

Git is one of the best ways to download Lithium, it allows you to easily pull the latest version of Lithium, just by typing `git pull` within the lithium directory. It will also allow you to download the complete history of lithium's source code, to download git or how to use it. See: http://git-scm.com/

You can use the following command to clone the git repository:

```
$ git clone git://github.com/kylef/lithium.git
```

## Getting the stable version

The latest development is actually stable, each change is tested before it is added to the development version. The difference is, that the stable version is a official release. The development version has lots of little changes. Once the amount of changes add up, it creates another stable release. The documentation is for the latest development release, but there are only slight differences. The documentation should apply to both releases.

### Via tar.gz / zip archive

The latest official build can be downloaded via:

- zip: http://cloud.github.com/downloads/kylef/lithium/lithium-0.2.zip

CHAPTER 4

# Contributing to Lithium

If you would like to help with Lithium, here you can find what needs to be done and how to contribute.

## Ideas

If you have any ideas on what could be added to Lithium, whether its a small template tag to an existing lithium app, or a new app. You can send your ideas to kylef.

## Code

If you have any patches for Lithium, you could send them to kylef and if the patches are good, he will add them. You could also fork the git repository, and send kylef a link to the git repository. Then kylef can keep track of what you are doing, and merge it back into the main repository if the changes are good.

Before making any changes or patches, please look closely at how lithium is written, such as 4 spaces instead of tabs. Follow the style of the code in your patches.

CHAPTER 5

# Base template

Most of, if not all lithium applications require a `base.html` template. The base template is just a master template on which every other template of lithium extends on. You can either use the provided templates, or write your own.

## Using the default templates

To use the default templates follow these steps:

1. Paste `from lithium import templates_path` into your `settings.py` file.
2. Add `templates_path()` to the `TEMPLATE_DIRS` section.

```
TEMPLATE_DIRS = (
    templates_path(),
)
```

## Writing your own base.html template

You will need to create a file called `base.html` and place it inside a TEMPLATE_DIR.

An example of a base.html template:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <title>{% block title %}My Website{% endblock %}</title>
</head>

<body>
    <div id="container">
```

```
        <div id="content">
            {% block content %}{% endblock %}
        </div>

        <div id="sidebar">
            ...
        </div>
    </div>
</body>
</html>
```

## What blocks does Lithium use?

Lithium only uses two blocks, title, and content. In the title, it will place the name of a page, the title of it. Inside the content, it will insert the bulk of the page, the information.

CHAPTER 6

# Lithium applications

The following documentation covers every application included with Lithium.

## Lithium applications

Here is a list of the applications which lithium provides.

### Lithium blog app

Lithium blog is a simple blog application which allows multi domain, multi user blogs.

**Note:** Lithium blog default templates require the `'django.contrib.comments'` application. But, if you write some custom templates you can work around this dependency.

#### Quick start guide

To get started using the `blog` app, follow these steps:

1. Install the blog app by adding `'lithium.blog'` to `INSTALLED_APPS`.
2. Run `manage.py syncdb` so that Django will create the post tables.
3. Add the blog app's URLs to your project's `urls.py`:

```
urlpatterns = patterns('',
        ...
        url(r'^blog/', include('lithium.blog.urls')),
        ...
)
```

### Template tags

Template tags are used within your blog's Templates to customize your blog, with the following templatetags you dynamically display posts or categories.

Like all custom template tag libraries, you'll need to load the custom tags before you can use them:

```
{% load blog %}
```

Once loaded you can use the template tags below.

### Get latest posts

This template tag will grab a list of the latest posts and insert them into the template context defined after the 'as' clause. Optionally, you can pass two other variables, a user, and a limit. The user clause will only display posts where the author is that user. The limit will limit the amount of posts in the list (the default is 5).

Syntax:

```
{% get_latest_posts as [varname] %}
{% get_latest_posts as [varname] limit [limit] %}
{% get_latest_posts for [user] as [varname] %}
{% get_latest_posts for [user] as [varname] limit [limit] %}
```

Example usage:

```
{% get_latest_posts as post_list limit 7 %}
{% for post in post_list %}
    {{ post.title }}
{% endfor %}
```

### Get category list

To get a list of categories you can use the `get_category_list` template tag. This tag will return a list of categories in alphabetical order. It will also filter the categories and only use categories where "favorite" is set. Like `get_latest_posts`, this tag also has a optional limit clause, except the default is 10.

Syntax:

```
{% get_category_list as [varname] %}
{% get_category_list as [varname] limit [limit] %}
```

Example usage:

```
{% get_category_list as category_list  %}
{% for category in category_list %}
    {{ post.tite }}
{% endfor %}
```

### Settings

Some of the application's behavior can be configured by adding the appropriate settings to your project's settings file.

#### BLOG_PAGINATE_BY

Default: 10

This is the amount of posts per page on any post list pages, such as the main post archive, tag post archive etc.

#### BLOG_PING

Default: False

When a post is created, if this is set to True, a post will be pinged to the servers in `BLOG_PING_SERVERS`.

#### BLOG_PING_SERVERS

Default:

```
BLOG_PING_SERVERS = (
    'http://blogsearch.google.com/ping/RPC2',
    'http://rpc.technorati.com/rpc/ping',
    'http://api.my.yahoo.com/rss/ping',
    'http://ping.feedburner.com',
)
```

This is a tuple of servers to be pinged.

#### BLOG_FEED_ITEMS

Default: 20

This is the amount of posts to display in the RSS Feeds for the blog.

### Templates

There are two templates which are used, these are blog/post_list.html and blog/post_detail.html.

#### blog/post_list.html

This template will get passed:

- `post_list`: The list of posts, this can be iterated over to display each Post within.
- `is_paginated`: A boolean representing whether the results are paginated. Specifically, this is set to False if the number of available posts is less than or equal to paginate_by.

If the results are paginated, the context will contain these extra variables:

- `paginator`: An instance of `django.core.paginator.Paginator`.
- `page_obj`: An instance of `django.core.paginator.Page`.

#### blog/post_detail.html

This template will get passed a `post` in the template context `post`.

## Lithium forum app

### Quick start guide

To get started using the `forum` app, follow these steps:

1. Install the blog app by adding `'lithium.forum'` to `INSTALLED_APPS`.
2. Run `manage.py syncdb` so that Django will create the forum tables.
3. Add the blog app's URLs to your project's `urls.py`:

```
urlpatterns = patterns('',
        ...
        url(r'^forum/', include('lithium.forum.urls')),
        ...
)
```

### Settings

Some of the application's behavior can be configured by adding the appropriate settings to your project's settings file.

#### FORUM_THREAD_PAGINATE_BY

Default: 40

This is the amount of threads per page on any thread list pages.

#### FORUM_POST_PAGINATE_BY

Default: 20

The amount of posts to display on a thread detail page.

#### FORUM_DEFAULT_READ_PERMISSION

Default: 1

The default read permission.

- 1 = Anonymous (Anonymous, User, Staff and Superuser may read thread or post inside this forum)
- 2 = User (User, Staff and Superuser may read a thread or post inside this forum)
- 3 = Staff (Staff and Superuser may read a thread or post inside this forum)
- 4 = Superuser (Only superusers may read a thread or post inside this forum)

#### FORUM_DEFAULT_WRITE_PERMISSION

Default: 2

The default write permission.

- 1 = Anonymous (Anonymous, User, Staff and Superuser may create a thread or post inside this forum)

- 2 = User (User, Staff and Superuser may create a thread or post inside this forum)
- 3 = Staff (Staff and Superuser may create a thread or post inside this forum)
- 4 = Superuser (Only superusers may create a thread or post inside this forum)

## Lithium contact app

Lithium contact is a simple contact form allowing users of your website to easily contact you via email.

### Quick start guide

To get started using the `contact` app, follow these steps:

1. Install the blog app by adding `'lithium.contact'` to `INSTALLED_APPS`.
2. Add the contact app's URLs to your project's `urls.py`:

```
urlpatterns = patterns('',
        ...
        url(r'^contact/', include('lithium.contact.urls')),
        ...
)
```

3. Add your email address to the `MANAGERS` section of `settings.py`:

```
MANAGERS = (
        ('Kyle Fuller', 'inbox@kylefuller.co.uk'),
)
```

### Using your own templates

#### `contact/contact_email.txt`

This template is used to render the email before it is sent. The template context will contain the following:

- `sender`: The sender's email address.
- `subject`: The sender's subject.
- `message`: The sender's message.
- `recipients`: This is a list of who the email is being sent to.

#### `contact/contact_form.html`

This template gets passed a `django.forms.Form` via the context `form`. This template should render the form, and display any validation errors if there were any.

### Rendering a custom form

A complete form might look like:

```
<form action="{% url contact_form %}" method="post">
    {{ form.as_p }}
    <input type="submit" value="Submit" />
</form>
```

#### contact/contact_confirmation.html

This template shows a confirmation to the user that there message has been sent.

## Lithium wiki app

### Quick start guide

To get started using the `wiki` app, follow these steps:

1. Install the wiki app by adding `'lithium.wiki'` to `INSTALLED_APPS`.
2. Run `manage.py syncdb` so that Django will create the wiki tables.
3. Add the wiki app's URLs to your project's `urls.py`:

```
urlpatterns = patterns('',
        ...
        url(r'^wiki/', include('lithium.wiki.urls')),
        ...
)
```

### Settings

Some of the application's behavior can be configured by adding the appropriate settings to your project's settings file.

#### WIKI_DEFAULT_WRITE_PERMISSION

Default: 1

This is the default permission required to write to a page. You can override this per page inside the administration panel. Here are a list of permissions you can use:

- 1 = Anonymous (Anonymouse, User, Staff and Superuser may edit)
- 2 = User (User, Staff and Superuser may edit)
- 3 = Staff (Staff and Superuser may edit)
- 4 = Superuser (Only superusers may edit a page)

#### WIKI_HISTORY_PAGINATE_BY

Default: 50

This is the amount of revisions per page on the page history.

### Templates

#### wiki/page_detail.html

This template will get passed a `Page` in the template context `page`.

#### wiki/page_discuss.html

This template works the same as `wiki/page_detail.html`, but it uses `django.contrib.comments` to add discussion to a page.

#### wiki/page_edit.html

This template get's passed a page, and a EditForm.

#### wiki/page_history.html

This template will get passed:

- `page`: The page which the revisions belong too.
- `revision_list`: The list of revisions, this can be iterated over to display each Revision within.
- `is_paginated`: A boolean representing whether the results are paginated. Specifically, this is set to False if the number of revisions is less than or equal to paginate_by.

If the results are paginated, the context will contain these extra variables:

- `paginator`: An instance of `django.core.paginator.Paginator`.
- `page_obj`: An instance of `django.core.paginator.Page`.

#### wiki/revision_detail.html

This template will get passed a `Revision` in the template context `revision`.

#### wiki/revision_diff.html

This template will get passed:

- `page`: The page to which each of these revisions belong to.
- `revisionA`: The first revision.
- `revisionB`: The second revision.
- `difftext`: The HTML difftext.

# Python Module Index

## l

# Index

## L