# CONTENTS

1 Features 3  
2 Installation 5  
3 Documentation 7  
4 License 9  
5 Contributing 11  
6 Contents 13  
  6.1 Getting Started ................................................................. 13  
  6.2 Using the Editor ................................................................ 15  
  6.3 Contributing Guide .............................................................. 17  
  6.4 Release History ................................................................. 19  
7 Indices and tables 21
django-scribbler is an application for managing snippets of text for a Django website. Similar projects include django-flatblocks, django-chunks and django-pagelets. This project attempts to take some of the best concepts from those previous projects as well as focus on giving the users instant feedback inspired by Bret Victor’s Inventing on Principle talk.
CHAPTER ONE

FEATURES

• Simple template tag for defining snippet blocks with default text
• Template tag for displaying and editing fields from arbitrary models
• Front-end editing of snippets with the powerful CodeMirror editor
• Live in-place preview of content while editing
• The full power of the Django template language in the snippet blocks
• Experimental Python 3 support

A very simple demo site is available at http://scribbler-mlavin.dotcloud.com/ (username: demo password: demo)
INSTALLED

django-scribbler requires Django >= 1.3 and Python >= 2.6. Starting with version v0.2 there will be experimental
Python 3 support (3.2+). Using Python 3 requires using the in-development version of Django (1.5dev).

To install from PyPi:

`pip install django-scribbler`
Documentation on using django-scribbler is available on Read The Docs.
django-scribbler is released under the BSD License. See the LICENSE file for more details.
CONTRIBUTING

If you think you’ve found a bug or are interested in contributing to this project check out django-scribbler on Github. A full contributing guide can be found in the online documentation

Development sponsored by Caktus Consulting Group, LLC.
6.1 Getting Started

Below are the basic steps need to get django-scribbler integrated into your Django project.

6.1.1 Configure Settings

You need to include scribbler to your installed apps. django-scribbler requires django.contrib.auth which in turn requires django.contrib.sessions which are enabled in Django by default. You will also need to include a context processor to include the current request in the template context.

```python
INSTALLED_APPS = (
    # Required contrib apps
    'django.contrib.auth',
    'django.contrib.sessions',
    # Other installed apps would go here
    'scribbler',
)

TEMPLATE_CONTEXT_PROCESSORS = (
    # Other context processors would go here
    'django.core.context_processors.request',
)
```

Note that TEMPLATE_CONTEXT_PROCESSORS is not included in the default settings created by startproject. You should take care to ensure that the default context processors are included in this list. For a list of default TEMPLATE_CONTEXT_PROCESSORS please see the official Django docs.

For the context processor to have any effect you need to make sure that the template is rendered using a RequestContext. This is done for you with the render shortcut.

django-scribbler aggressively caches the scribble content. By default the scribble content is cached for 12 hours. You have the option to configure this cache timeout with the SCRIBBLER_CACHE_TIMEOUT setting. The value should be the timeout in seconds.

6.1.2 Configure Urls

You should include the scribbler urls in your root url patterns.
urlpatterns = patterns(‘’,
    # Other url patterns would go here
    url(r’^scribbler/’, include(‘scribbler.urls’)),
)

6.1.3 Create Database Tables

You’ll need to create the necessary database tables for storing scribble content. This is done with the `syncdb` management command built into Django:

```python
python manage.py syncdb
```

django-scribbler uses South to handle database migrations. If you are also using South then you should run `migrate` instead:

```python
python manage.py migrate scribbler
```

**Note:** The latest release of South does not support Python 3. If you want to try django-scribbler with Python 3 you will have go without South for the time being.

6.1.4 User Permissions

To edit scribbles on the front-end users must have the `scribbler.add_scribble` and `scribbler.change_scribble` permissions. You can configure users to have these permissions through the users section of the Django admin. Superusers have all of these permissions by default.

Similarly, to edit fields from models on the front-end, users must have “change” permission for the models being edited. Again these permissions can be configured through the users section of the Django admin.

6.1.5 Include Static Resources

django-scribbler includes both CSS and JS resources which need to be included in your templates to handle the front-end content management. Since you may want to include scribbles on any page on your site these should be included in your base template `<head>`.

```html
<link rel="stylesheet" href="{{ STATIC_URL }}scribbler/libs/codemirror.css">
<link rel="stylesheet" href="{{ STATIC_URL }}scribbler/css/scribbler.css">
<script data-main="{{ STATIC_URL }}scribbler/js/scribbler" src="{{ STATIC_URL }}scribbler/libs/require.js"></script>
```

This uses RequireJS to load the additional JS resources. The front-end editor uses CodeMirror (currently using v2.32) which is included in the distribution. Both RequireJS and CodeMirror are available a MIT-style license compatible with this project’s BSD license. You can find the license files included in scribbler/static/scribbler/libs/.

6.1.6 Place Scribbles in Your Template

You are now ready to place the scribble content blocks throughout your templates. This is done with the `scribble` block tag. It takes one argument which is the slug name for the scribble. Slugs must be unique per url/slug pair. That means you cannot use the same slug more than once in the template but you can use the same slug in different templates as long as they are rendered on different urls.
{% load scribbler_tags %}
{% scribble 'header' %}

<p>Blip {{ now 'Y' }} {{ STATIC_URL|upper }}</p>
{% endscribble %}

The content inside the block is the default content that will be rendered if a matching scribble in the database is not found.

**Note:** Scribble content can be any valid Django template. However the content does not include all of the context of the template. Only the context provided by the set of `TEMPLATE_CONTEXT_PROCESSORS`.

A second scribbler tag, `scribble_field`, allows for editing fields of model instances. For example, suppose you have a `DaysLog` model with a field named happenings. Suppose an instance of this model is passed into your template in the template variable `days_log`. Then the happenings field of this `DaysLog` instance can be displayed and edited on the page by including this `scribble_field` template tag in the template for the page:

{% load scribbler_tags %}
{% scribble_field days_log 'happenings' %}

**Note:** The logged-in user must have “change” permission for the model in order for the model instance to be editable on the page.

That should be enough to get you up and running with django-scribbler.

### 6.2 Using the Editor

django-scribbler makes use of CodeMirror to create a powerful client-side editor. We’ve added a couple features to make it easier when working with Django templates.

#### 6.2.1 Context Inspection

When using the editor, you can inspect the current context by starting a variable node with `{%` and hitting tab. As noted in the quick start introduction, scribble content can be any valid Django template. The context provided when rendering the scribble includes anything added by the set of `TEMPLATE_CONTEXT_PROCESSORS`. This would include `STATIC_URL`, `MEDIA_URL`, `LANGUAGE_CODE`, the current user and others. Developers may choose to add context processors to include additional content for rendering scribbles.
6.2.2 Template Tag/Filter Completion

Similar to how the editor can tab-complete the context variables, you can tab complete template tags when `{%` has been opened. The built-in filters can be tab-completed when the pipe `|` character is detected inside of a variable node. Currently this will only complete the built-in tags and filter and will not include any additional tags or filters which might be added by loading additional libraries inside the scribble.

6.2.3 Saving Drafts

While editing the scribble content, the editor will periodically save the current editor content as a draft. These drafts are saved on the client side using local storage (if supported by the browser) or a cookie. While that means you won’t be able to see or use these drafts in another browser, it does mean that you work will not be lost if there is a failure on the server while saving changes or making edits. When the editor is opened it will restore any draft that is found.
6.3 Contributing Guide

There are a number of ways to contribute to django-scribbler. If you are interested in making django-scribbler better then this guide will help you find a way to contribute.

6.3.1 Ways to Contribute

You can contribute to the project by submitting bug reports, feature requests or documentation updates through the Github issues.

6.3.2 Getting the Source

You can clone the repository from Github:

```
git clone git://github.com/caktus/django-scribbler.git
```

However this checkout will be read only. If you want to contribute code you should create a fork and clone your fork. You can then add the main repository as a remote:

```
git clone git@github.com:<your-username>/scribbler.git
git remote add upstream git://github.com/caktus/django-scribbler.git
git fetch upstream
```

6.3.3 Running the Tests

When making changes to the code, either fixing bugs or adding features, you’ll want to run the tests to ensure that you have not broken any of the existing functionality. With the code checked out and Django installed you can run the tests via:

```
python setup.py test
```

or:

```
python runtests.py
```

Note that the tests require the mock library. To test against multiple versions of Django you can use install and use tox>=1.4. The tox command will run the tests against Django 1.3, 1.4 and the current Git master using Python 2.6:

```
# Build all environments
 tox
# Build a single environment
 tox -e py26-1.3.X
```

Building all environments will also build the documentation. More on that in the next section.

6.3.4 Building the Documentation

This project aims to have a minimal core with hooks for customization. That makes documentation an important part of the project. Useful examples and notes on common use cases are a great way to contribute and improve the documentation.
The docs are written in ReST and built using Sphinx. As noted above you can use tox to build the documentation or you can build them on their own via:

    tox -e docs

or:

    make html

from inside the docs/ directory.

### 6.3.5 Building the CSS

The CSS used by django-scribbler is built using LESS. No changes should be made to the scribbler.css directly. Instead changes should be made to the scribbler.less file. After changes are made to scribbler.less you can create the new CSS with the Node based complier:

    # Install less from the NPM package
    npm install less -g
    lessc scribbler/static/scribbler/less/scribbler.less > scribbler/static/scribbler/css/scribbler.css

The example project uses the client-side LESS compiler to make local development easier.

### 6.3.6 Coding Standards

Code contributions should follow the PEP8 and Django contributing style standards. Please note that these are only guidelines. Overall code consistency and readability are more important than strict adherence to these guides.

The Javascript is configured for some basic JSHint checks. Changes to the Javascript should pass without errors. You can check the Javascript file on the command line with Node based CLI tool:

    # Install jshint from the NPM package
    npm install jshint -g
    # Check the scribbler JS
    jshint scribbler/static/scribbler/js/scribbler.js

### 6.3.7 Submitting a Pull Request

The easiest way to contribute code or documentation changes is through a pull request. For information on submitting a pull request you can read the Github help page [https://help.github.com/articles/using-pull-requests](https://help.github.com/articles/using-pull-requests).

Pull requests are a place for the code to be reviewed before it is merged. This review will go over the coding style as well as if it solves the problem intended and fits in the scope of the project. It may be a long discussion or it might just be a simple thank you.

Not necessarily every request will be merged but you should not take it personally if your change is not accepted. If you want to increase the chances of your change being incorporated then here are some tips.

- Address a known issue. Preference is given to a request that fixes a currently open issue.
- Include documentation and tests when appropriate. New features should be tested and documented. Bugfixes should include tests which demostrate the problem.
- Keep it simple. It’s difficult to review a large block of code so try to keep the scope of the change small.

You should also feel free to ask for help writing tests or writing documentation if you aren’t sure how to go about it.
6.4 Release History

Release and change history for django-scribbler

6.4.1 v0.3.0 (Released 2012-10-26)

Features

• Autocomplete for Django template tags and filters
• New scribble_field template tag to allow editing of fields in arbitrary models

6.4.2 v0.2.1 (Released 2012-10-12)

Bug Fixes

• Preview was broken when scribble was saved due to unique constraint. See #34

6.4.3 v0.2.0 (Released 2012-10-12)

The editor now saves drafts on the client side by default. Python 3 support is added when using the lastest Django master. There is also some additional documentation.

A unique constraint was added and upgrading from v0.1 does require a migration:

```
manage.py migrate scribbler
```

• Added experimental Python >= 3.2 support when using Django 1.5dev
• Caktus Consulting Group has taken over the primary development
• Added the ability to save as a draft on the client side
• Added an official contributing guide

Bug Fixes

• Added unique constraint for url/slug pair. South migration is included.

6.4.4 v0.1.1 (Released 2012-08-25)

Minor bug fix release for some JS and CSS issues.

Bug Fixes

• Fixed issue with the content editor z-index allowing content in front when open
• Fixed issue where links within editable content could not be clicked by editors
6.4.5 v0.1.0 (Released 2012-07-28)

- Initial public release.

Features

- Template tag for rendering content blocks
- CodeMirror editor integration
CHAPTER
SEVEN

INDICES AND TABLES

• genindex
• modindex
• search