django-fluent-comments Documentation

Release 3.0

Diederik van der Boor

May 11, 2021

Contents:

The *django-fluent-comments* module enhances the default appearance of the django_comments application to be directly usable in web sites.

Features:

- Ajax-based preview and posting of comments
- Configurable and flexible form layouts.
- Comment moderation, with auto-closing / auto-moderation after N days.
- E-mail notification to the site managers of new comments.
- Optional threaded comments support via django-threadedcomments.
- Optional Akismet integration for spam detection.
- Optional reCAPTCHA2 support via django-recaptcha or django-nocaptcha-recaptcha.
- Optional simple captcha support via django-simple-captcha.

The application is designed to be plug&play; installing it should already give a better comment layout.

CHAPTER 1

Installation

First install the module and django_comments, preferably in a virtual environment:

```
pip install django-fluent-comments
```

1.1 Configuration

To use comments, the following settings are required:

```
INSTALLED_APPS += (
    'fluent_comments',  # must be before django_comments
    'crispy_forms',
    'django_comments',
    'django.contrib.sites',
)

CRISPY_TEMPLATE_PACK = 'bootstrap3'

COMMENTS_APP = 'fluent_comments'
```

Add the following in `urls.py`:

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

The database can be created afterwards:

```
./manage.py migrate
```

1.2 Usage in the page

Provide a template that displays the comments for the `object` and includes the required static files:

```
{% load comments static %}

<link rel="stylesheet" type="text/css" href="{% static 'fluent_comments/css/
↪ajaxcomments.css' %}" />
<script type="text/javascript" src="{% static 'fluent_comments/js/ajaxcomments.js' %}
↪"></script>

{% render_comment_list for object %}
{% render_comment_form for object %}
```

Note: When using the comment module via django-fluent-contents or django-fluent-blogs, this step can be omitted.

1.3 Template for non-ajax pages

The templates which django_comments renders use a single base template for all layouts. This template is empty by default since it's only serves as a placeholder. To complete the configuration of the comments module, create a `comments/base.html` file that maps the template blocks onto your website base template. For example:

```
{% extends "mysite/base.html" %}{% load i18n %}

{% block meta-title %}{% block title %}{% trans "Responses for page" %}{% endblock %}{
↪% endblock %}

{% block main %}
    <div id="comments-wrapper">
        {% block content %}{% endblock %}
    </div>
{% endblock %}
```

In this example, the base template has a `meta-title` and `main` block, which contain the `content` and `title` blocks that django_comments needs to see. This application also outputs an `extrahead` block for a meta-refresh tag. The `extrahead` block can be included in the site base template directly, so it doesn't have to be included in the `comments/base.html` file.

CHAPTER 2

Changing the form layout

Form layouts generally differ across web sites, hence this application doesn't dictate a specific form layout. Instead, this application uses django-crispy-forms which allows configuration of the form appearance.

The defaults are set to Bootstrap 3 layouts, but can be changed.

For example, use:

```
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

2.1 Using a different form class

By choosing a different form class, the form layout can be redefined at once:

The default is:

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.DefaultCommentForm'

FLUENT_COMMENTS_FORM_CSS_CLASS = 'comments-form form-horizontal'
FLUENT_COMMENTS_LABEL_CSS_CLASS = 'col-sm-2'
FLUENT_COMMENTS_FIELD_CSS_CLASS = 'col-sm-10'
```

You can replace the labels with placeholders using:

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.CompactLabelsCommentForm'
```

Or place some fields at a single row:

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.CompactCommentForm'

# Optional settings for the compact style:
FLUENT_COMMENTS_COMPACT_FIELDS = ('name', 'email', 'url')
FLUENT_COMMENTS_COMPACT_GRID_SIZE = 12
FLUENT_COMMENTS_COMPACT_COLUMN_CSS_CLASS = "col-sm-{size}"
```

2.2 Changing the field order

The default is:

```
FLUENT_COMMENTS_FIELD_ORDER = ('name', 'email', 'url', 'comment')
```

For a more modern look, consider placing the comment first:

```
FLUENT_COMMENTS_FIELD_ORDER = ('comment', 'name', 'email', 'url')
```

2.3 Hiding form fields

Form fields can be hidden using the following settings:

```
FLUENT_COMMENTS_EXCLUDE_FIELDS = ('name', 'email', 'url')
```

When *django-threadedcomments* are used, the `title` field can also be removed.

Note: Omitting fields from `FLUENT_COMMENTS_FIELD_ORDER` has the same effect.

2.4 Using a custom form class

When the settings above don't provide the layout you need, you can define a custom form class entirely:

```
from fluent_comments.forms import CompactLabelsCommentForm

# Or for recaptcha as base, import:
from fluent_comments.forms.recaptcha import CompactCommentForm


class CommentForm(CompactLabelsCommentForm):
    """
    The comment form to use
    """

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields['url'].label = _("Website")  # Changed the label
        self.fields['email'].label = _("Email address (will not be published)")
```

And use that class in the `FLUENT_COMMENTS_FORM_CLASS` setting. The `helper` attribute defines how the layout is constructed by django-crispy-forms, and should be redefined the change the field ordering or appearance.

2.5 Switching form templates

By default, the forms can be rendered with 2 well known CSS frameworks:

- Bootstrap The default template pack. The popular simple and flexible HTML, CSS, and Javascript for user interfaces from Twitter.

- Uni-form Nice looking, well structured, highly customizable, accessible and usable forms.

The `CRISPY_TEMPLATE_PACK` setting can be used to switch between both layouts. For more information, see the django-crispy-forms documentation.

Both CSS frameworks have a wide range of themes available, which should give a good head-start to have a good form layout. In fact, we would encourage to adopt django-crispy-forms for all your applications to have a consistent layout across all your Django forms.

If your form CSS framework is not supported, you can create a template pack for it and submit a pull request to the django-crispy-forms authors for inclusion.

CHAPTER 3

Using custom comment templates

Besides the standard templates of django-comments, this package provides a `comments/comment.html` template to render a single comment.

It's default looks like:

```
{% load i18n %}
<div{% if preview %} id="comment-preview"{% else %} id="c{{ comment.id }}"{% endif %}
↪class="comment-item">
 {% block comment_item %}
   {% if preview %}<h3>{% trans "Preview of your comment" %}</h3>{% endif %}
     <h4>
       {% block comment_title %}
         {% if comment.url %}<a href="{{ comment.url }}" rel="nofollow">{% endif %}
         {% if comment.name %}{{ comment.name }}{% else %}{% trans "Anonymous" %}{%
↪endif %}{% comment %}
         {% endcomment %}{% if comment.url %}</a>{% endif %}
         <span class="comment-date">{% blocktrans with submit_date=comment.submit_
↪date %}on {{ submit_date }}{% endblocktrans %}</span>
         {% if not comment.is_public %}<span class="comment-moderated-flag">({% trans
↪"moderated" %})</span>{% endif %}
         {% if USE_THREADEDCOMMENTS and not preview %}<a href="#c{{ comment.id }}"
↪data-comment-id="{{ comment.id }}" class="comment-reply-link">{% trans "reply" %}</
↪a>{% endif %}
       {% endblock %}
     </h4>

     <div class="comment-text">{{ comment.comment|linebreaks }}</div>
 {% endblock %}
</div>
```

Note: The `id="comment-preview"`, `data-comment-id` fields are required for proper JavaScript actions. The div id should be `id="c{{ comment.id }}"`, because `Comment.get_absolute_url()` points to it.

Adding a Bootstrap 4 layout, including Gravatar would look like:

```
{% load i18n gravatar %}

<div id="{% if preview %}comment-preview{% else %}c{{ comment.id }}{% endif %}" class=
↪"comment-item{% if comment.user_id and comment.user_id == comment.content_object.
↪author_id %} by-author{% endif %}">
  {% if preview %}<h3>{% trans "Preview of your comment" %}</h3>{% endif %}
  <div class="media">
    {% gravatar comment.email css_class='user-image' %}
    <div class="media-body">
      <h4>
        {% block comment_title %}
          {% if comment.url %}<a href="{{ comment.url }}" rel="nofollow">{% endif %}
          {% if comment.name %}{{ comment.name }}{% else %}{% trans "Anonymous" %}{%␣
↪endif %}{% comment %}
          {% endcomment %}{% if comment.url %}</a>{% endif %}
          {% if not comment.is_public %}<span class="comment-moderated-flag">({%␣
↪trans "moderated" %})</span>{% endif %}
          {% if comment.user_id and comment.user_id == comment.content_object.author_
↪id %}<span class="comment-author-flag">[{% trans "author" %}]</span>{% endif %}
        {% endblock %}
      </h4>

      <div class="comment-text">{{ comment.comment|linebreaks }}</div>
      <div class="comment-tools">
        {% if USE_THREADEDCOMMENTS and not preview %}<a href="#c{{ comment.id }}"␣
↪data-comment-id="{{ comment.id }}" class="comment-reply-link">{% trans "reply" %}</
↪a>{% endif %}
        <span class="comment-date">{{ comment.submit_date }}</span>
      </div>
    </div>
  </div>
</div>
```

Warning: While extremely popular, Gravatar is a huge privacy risk, as it acts like a tracking-pixel for all your users. It also exposes email addresses as the MD5 hashes can be reverse engineerd. See the *GDPR* notes for more information.

3.1 Customize date time formatting

To override the displayed date format, the template doesn't have to be overwritten. Instead, define `DATETIME_FORMAT` in a locale file. Define the following setting:

```
FORMAT_MODULE_PATH = 'settings.locale'
```

Then create `settings/locale/XY/formats.py` with:

```
DATETIME_FORMAT = '...'
```

This should give you consistent dates across all views.

CHAPTER 4

Captcha support

Users can be required to enter a captcha.

This is done by changing the *FLUENT_COMMENTS_FORM_CLASS* setting.

Note: When *FLUENT_COMMENTS_FIELD_ORDER* is configured, also include the `"captcha"` field!

4.1 Using django-recaptcha

django-recaptcha provides "no captcha" reCAPTCHA v2 support. Choose one of the form layout classes:

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.recaptcha.DefaultCommentForm'  #␣
↪default
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.recaptcha.CompactLabelsCommentForm
↪'  # no labels
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.recaptcha.CompactCommentForm'  #␣
↪compact row
```

And configure it's settings:

```
RECAPTCHA_PUBLIC_KEY = "the Google provided site_key"
RECAPTCHA_PRIVATE_KEY = "the Google provided secret_key"

NOCAPTCHA = True  # Important! Required to get "no captcha" reCAPTCHA v2

INSTALLED_APPS += (
    'captcha',
)
```

4.2 Using django-nocaptcha-recaptcha

django-nocaptcha-recaptcha also provides "no captcha" reCAPTCHA v2 support. The same form classes are used, as the correct imports are detected at startup:

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.recaptcha.DefaultCommentForm'  #␣
↪default
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.recaptcha.CompactLabelsCommentForm
↪'  # no labels
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.recaptcha.CompactCommentForm'  #␣
↪compact row
```

It's settings differ slightly:

```
NORECAPTCHA_SITE_KEY = "the Google provided site_key"
NORECAPTCHA_SECRET_KEY = "the Google provided secret_key"

INSTALLED_APPS += (
    'nocaptcha_recaptcha',
)
```

4.3 Using django-simple-captcha

django-simple-captcha provides a simple local captcha test. It does not require external services, but it can be easier to break.

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.captcha.DefaultCommentForm'  #␣
↪default
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.captcha.CompactLabelsCommentForm'␣
↪ # no labels
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.captcha.CompactCommentForm'  #␣
↪compact row
```

And configure the app:

```
CAPTCHA_NOISE_FUNCTIONS = ()
CAPTCHA_FONT_SIZE = 30
CAPTCHA_LETTER_ROTATION = (-10,10)

INSTALLED_APPS += (
    'captcha',
)
```

Warning: Note that both django-simple-captcha and django-recaptcha use the same "captcha" module name. These packages can't be installed together.

CHAPTER 5

Akismet spam detection

Akismet is used out of the box when the `AKISMET_API_KEY` setting is defined:

```
AKISMET_API_KEY = "your-api-key"
```

This can also be enabled explicitly:

```
FLUENT_CONTENTS_USE_AKISMET = True  # Enabled by default when AKISMET_API_KEY is set.
```

The following settings are optional:

```
AKISMET_BLOG_URL = "http://example.com"  # Optional, to override auto detection
AKISMET_IS_TEST = False  # Enable to make test runs
```

When spam is detected, the default behavior depends on the spam score. Obvious spam is discarded with an HTTP 400 response, while possible spam is marked for moderation.

The *FLUENT_COMMENTS_AKISMET_ACTION* setting can be one of these values:

- `auto` chooses between `moderate`, `soft_delete` and `delete` based on the spam score.
- `moderate` will always mark the comment for moderation.
- `soft_delete` will mark the comment as moderated + removed, but it can still be seen in the admin.
- `delete` will outright reject posting the comment and respond with a HTTP 400 Bad Request.

Tip: By default, Akismet will not report any post from the Django superuser as spam. Comments with the name "viagra-test-123" will always be flagged as spam.

Warning: Akismet is a third party service by Automattic. Note that *GDPR Compliance* is next-to-impossible with this service.

CHAPTER 6

E-mail notification

By default, the MANAGERS of a Django site will receive an e-mail notification of new comments. This feature can be enabled or disabled using:

```
FLUENT_COMMENTS_USE_EMAIL_NOTIFICATION = False
```

By default, plain-text e-mail messages are generated using the template `comments/comment_notification_email.txt`.

Multi-part (HTML) e-mails are supported using the template `comments/comment_notification_email.html`. To enabled multi-part e-mails, set:

```
FLUENT_COMMENTS_MULTIPART_EMAILS = True
```

In addition to the standard django-comments package, the `request` and `site` fields are available in the template context data. This allows generating absolute URLs to the site.

CHAPTER 7

Auto comment moderation

By default, any comment receives moderation from the "default moderator" This ensures random comments also receive *akismet* checks, bad word filtering and send *email notifications*.

Some moderation features require more knowledge of the model. This includes:

- Toggling an "enable comments" checkbox on the model.
- Auto-closing comments after X days since the publication of the article.
- Auto-moderating comments after X days since the publication of the article.

Comment moderation can be enabled for the specific models using:

```
from fluent_comments.moderation import moderate_model
from myblog.models import BlogPost

moderate_model(BlogPost,
    publication_date_field='publication_date',
    enable_comments_field='enable_comments',
)
```

This code can be placed in a `models.py` file. The provided field names are optional. By providing the field names, the comments can be auto-moderated or auto-closed after a number of days since the publication date.

The following settings are available for comment moderation:

```
FLUENT_COMMENTS_CLOSE_AFTER_DAYS = None          # Auto-close comments after N days
FLUENT_COMMENTS_MODERATE_AFTER_DAYS = None       # Auto-moderate comments after N days.
```

7.1 The default moderator

The default moderator is configurable using:

```
FLUENT_COMMENTS_DEFAULT_MODERATOR = 'default'
```

Possible values are:

- `default` installs the standard moderator that all packages use.
- `deny` will reject all comments placed on models which don't have an explicit moderator registered with `moderate_model()`.
- `None` will accept all comments, as if there is no default moderator installed.
- A dotted Python path will import this class.

When using a custom moderator class, consider inheriting `fluent_comments.moderation.FluentCommentsModerator` to preserve the email notification feature.

CHAPTER 8

Adding threaded comments

This package has build-in support for django-threadedcomments in this module. It can be enabled using the following settings:

```
INSTALLED_APPS += (
    'threadedcomments',
)

COMMENTS_APP = 'fluent_comments'
```

And make sure the intermediate `ThreadedComment` model is available and filled with data:

```
./manage.py migrate
./manage.py migrate_comments
```

The templates and admin interface adapt themselves automatically to show the threaded comments.

CHAPTER 9

IP-Address detection

This package stores the remote IP of the visitor in the model, and passes it to *Akismet* for spam detection. The IP Address is read from the `REMOTE_ADDR` meta field. In case your site is behind a HTTP proxy (e.g. using Gunicorn or a load balancer), this would make all comments appear to be posted from the load balancer IP.

The best and most secure way to fix this, is using WsgiUnproxy middleware in your `wsgi.py`:

```
from django.core.wsgi import get_wsgi_application
from django.conf import settings
from wsgiunproxy import unproxy

application = get_wsgi_application()
application = unproxy(trusted_proxies=settings.TRUSTED_X_FORWARDED_FOR_
↪IPS)(application)
```

In your `settings.py`, you can define which hosts may pass the `X-Forwarded-For` header in the HTTP request. For example:

```
TRUSTED_X_FORWARDED_FOR_IPS = (
    '11.22.33.44',
    '192.168.0.1',
)
```

Warning: Please don't try to read `HTTP_X_FORWARDED_FOR` blindly with a fallback to `HTTP_REMOTE_ADDR`. These headers could be provided by hackers, effectively circumventing your IP-address checks. Use WsgiUnproxy instead, which protects against maliciously injected headers.

9.1 Amazon Web Services Support

For AWS hosting, there is also wsgi-aws-unproxy which does the same for all CloudFront IP addresses.

9.2 IP-Subnet filtering

Use the `netaddr` package to trust a full IP-block, e.g. for Kubernetes Ingress:

```
from django.core.wsgi import get_wsgi_application
from netaddr import IPNetwork
from wsgiunproxy import unproxy

application = get_wsgi_application()
application = unproxy(trusted_proxies=IPNetwork('10.0.0.0/8'))(application)
```

CHAPTER 10

Privacy concerns (GDPR)

Comment support needs to consider the General Data Protection Regulation (GDPR) when when you serve European customers. Any personal data (email address, IP-address) should only be stored as long as this is truely needed, and it must be clear whom it's shared with.

Tip: For a simple introduction, see https://premium.wpmudev.org/blog/gdpr-compliance/

The Django comments model also stores the email address and IP-address of the commenter, which counts as personal information a user should give consent for. Consider running a background task that removes the IP-address or email address after a certain period.

10.1 Concerns for third-party services

When using *Akismet*, the comment data and IP-address is passed to the servers of Akismet.

In case you update templates to display user avatars using Gravatar, this this also provides privacy-sensitive information to a third party. Gravatar acts like a tracking-pixel, noticing every place you visit. It also makes your user's email address public. While the URL field is encoded as MD5, Gravatar doesn't use salted hashes so the data can be easily reverse engineered back to real user accounts.

See also:

For more information, read:

- https://meta.stackexchange.com/questions/21117/is-using-gravatar-a-security-risk
- https://webapps.stackexchange.com/questions/9973/is-it-safe-to-use-gravatar/30605#30605
- http://onemansblog.com/2007/02/02/protect-your-privacy-delete-internet-usage-tracks/comment-page-1/#comment-46204
- https://www.wordfence.com/blog/2016/12/gravatar-advisory-protect-email-address-identity/

CHAPTER 11

Configuration reference

The default settings are:

```
AKISMET_API_KEY = None
AKISMET_BLOG_URL = None
AKISMET_IS_TEST = False

CRISPY_TEMPLATE_PACK = 'bootstrap'

FLUENT_COMMENTS_REPLACE_ADMIN = True

# Akismet spam fighting
FLUENT_CONTENTS_USE_AKISMET = bool(AKISMET_API_KEY)
FLUENT_COMMENTS_AKISMET_ACTION = 'soft_delete'

# Moderation
FLUENT_COMMENTS_DEFAULT_MODERATOR = 'default'
FLUENT_COMMENTS_CLOSE_AFTER_DAYS = None
FLUENT_COMMENTS_MODERATE_BAD_WORDS = ()
FLUENT_COMMENTS_MODERATE_AFTER_DAYS = None
FLUENT_COMMENTS_USE_EMAIL_NOTIFICATION = True
FLUENT_COMMENTS_MULTIPART_EMAILS = False

# Form layouts
FLUENT_COMMENTS_FIELD_ORDER = ()
FLUENT_COMMENTS_EXCLUDE_FIELDS = ()
FLUENT_COMMENTS_FORM_CLASS = None
FLUENT_COMMENTS_FORM_CSS_CLASS = 'comments-form form-horizontal'
FLUENT_COMMENTS_LABEL_CSS_CLASS = 'col-sm-2'
FLUENT_COMMENTS_FIELD_CSS_CLASS = 'col-sm-10'

# Compact style settings
FLUENT_COMMENTS_COMPACT_FIELDS = ('name', 'email', 'url')
FLUENT_COMMENTS_COMPACT_GRID_SIZE = 12
FLUENT_COMMENTS_COMPACT_COLUMN_CSS_CLASS = "col-sm-{size}"
```

11.1 FLUENT_COMMENTS_FORM_CLASS

Defines a dotted Python path to the form class to use. The built-in options include:

Standard forms:

- `fluent_comments.forms.DefaultCommentForm` The standard form.
- `fluent_comments.forms.CompactLabelsCommentForm` A form where labels are hidden.
- `fluent_comments.forms.CompactCommentForm` A compact row

Variations with reCAPTCHA v2:

- `fluent_comments.forms.recaptcha.DefaultCommentForm`
- `fluent_comments.forms.recaptcha.CompactLabelsCommentForm`
- `fluent_comments.forms.recaptcha.CompactCommentForm`

Variations wiwth a simple self-hosted captcha:

- `fluent_comments.forms.captcha.DefaultCommentForm`
- `fluent_comments.forms.captcha.CompactLabelsCommentForm`
- `fluent_comments.forms.captcha.CompactCommentForm`

11.2 FLUENT_COMMENTS_AKISMET_ACTION

What to do when spam is detected, see *Akismet spam detection*.

11.3 FLUENT_COMMENTS_FIELD_ORDER

Defines the field ordering, see *Changing the field order*.

CHAPTER 12

Changelog

12.1 Version 3.0 (2021-05-11)

- Added Django 3 compatibility.
- Added HTML email support (`FLUENT_COMMENTS_MULTIPART_EMAILS = True` setting)
- Fix duplicated comment forms in threaded response.
- Drop Django 1.8, 1.9 and 1.10 compatibility.
- Drop Python 2 support.

12.2 Version 2.1 (2018-08-27)

- Make sure comment moderation is always active.
- Added a "default moderator" for the models that are not registered via `moderate_model()`.
- The default moderator is configurable via `FLUENT_COMMENTS_DEFAULT_MODERATOR`.
- Spam filtering works, but "auto close/moderate after" support needs a registration via `moderate_model()`.
- Added simple captcha support.
- Added "no captcha" reCAPTCHA2 support.
- Add new default `FLUENT_COMMENTS_AKISMET_ACTION=auto` option that completely discards comments when Akismet classifies as definitive spam.
- Fixed using `force_text()` to generate the content object title for email.
- Fixed showing HTML in the comments admin.
- Fixed showing the preview multiple times for threaded comments.
- Included `form.is_preview` flag.

12.3 Version 2.0.2 (2018-05-08)

- Fixed comment moderation when django-threadedcomments was used.

12.4 Version 2.0.1 (2018-05-04)

- Fixed migration file.
- Fixed missing Dutch translations.
- Improved default form button labels.

12.5 Version 2.0 (2018-01-22)

- Added Django 2.0 support.
- Dropped Django 1.5 / 1.6 / 1.7 support.
- Dropped Python 2.6 support.
- Dropped django.contrib.comments support.

12.6 Version 1.4.3 (2017-08-16)

- Fixed the IP-address reported in the email notification, the database records stored the actual correct value.
- Fixed missing `request` variable in templates.
- Fixed wrapping of the `ThreadedComment` model by the `FluentComment` proxy model too.

12.7 Version 1.4.2 (2017-07-08)

- Fixed Django 1.11 appearance of compact labels; e-mail and URL field didn't receive a placeholder anymore.
- Fixed HTML position of the hidden `parent` field.
- Enforce python-akismet >= 0.3 for Python 3 compatibility.

12.8 Version 1.4.1 (2017-02-06)

- Fixed compatibility with django_comments 1.8.

12.9 Version 1.4 (2017-02-03)

- Added `fluent_comments.forms.CompactLabelsCommentForm` style for `FLUENT_COMMENTS_FORM_CLASS`.

- Added `FLUENT_COMMENTS_MODERATE_BAD_WORDS` setting, to auto moderate on profanity or spammy words.
- Added `FLUENT_COMMENTS_AKISMET_ACTION = "soft_delete"` to auto-remove spammy comments. This is now the new default too.
- Exposed all form styles through `fluent_comments.forms` now.
- Fixed `is_superuser` check in moderation.
- Fixed `blog_language` parameter for Akismet.

12.10 Version 1.3 (2017-01-02)

- Added Akismet support for Python 3, via python-akismet.
- Added field reordering support, via the `FLUENT_COMMENTS_FIELD_ORDER` setting.
- Added form class swapping, through the `FLUENT_COMMENTS_FORM_CLASS` setting.
- Added new compact-form style, enable using:

```
FLUENT_COMMENTS_FORM_CLASS = 'fluent_comments.forms.CompactCommentForm'
FLUENT_COMMENTS_COMPACT_FIELDS = ('name', 'email', 'url')
```

- Added template blocks to override `comments/form.html` via `comments/app_name/app_label/form.html`.
- Added support for `app_name/app_label` template overrides to our `comments/comment.html` template.

12.11 Version 1.2.2 (2016-08-29)

- Allow non-integer primary key
- Added Slovak translation

12.12 Version 1.2.1 (2016-05-23)

- Fixed error handling in JavaScript when server reports an error.

12.13 Version 1.2 (2015-02-03)

- Fixed Django 1.9 support.

12.14 Version 1.1 (2015-12-28)

- Fix Django 1.9 issue with imports.
- Fix error in the admin for non-existing objects.

- Fix Python 3 installation error (dropped Akismet requirement).
- Drop Django 1.4 compatibility (in the templates).

12.15 Version 1.0.5 (2015-10-17)

- Fix Django 1.9 issue with importing models in `__init__.py`.
- Fix django-threadedcomments 1.0.1 support

12.16 Version 1.0.4 (2015-10-01)

- Fixed `get_comments_model()` import.

12.17 Version 1.0.3 (2015-09-01)

- Fix support for `TEMPLATE_STRING_IF_INVALID`, avoid parsing the "for" argument in `{% ajax_comment_tags for object %}`.
- Look for the correct `#id_parent` node (in case there are multiple)
- Improve Bootstrap 3 appearance (template can be overwritten).

12.18 Version 1.0.2

- Fixed packaging bug

12.19 Version 1.0.1

- Fix app registry errors in Django 1.7
- Fix security hash formatting errors on bad requests.

12.20 Version 1.0.0

- Added Django 1.8 support, can use either the django_comments or the django.contrib.comments package now.
- Fixed Python 3 issue in the admin
- Fixed unicode support in for subject of notification email

12.20.1 Released as 1.0b1

- Fixed ajax-comment-busy check
- Fixed clearing the whole container on adding comment

12.20.2 Released as 1.0a2

- Fix installation at Python 2.6

12.20.3 Released as 1.0a1

- Added support for Python 3 (with the exception of Akismet support).
- Added support for multiple comment area's in the same page.

  NOTE: any custom templates need to be updated, to use the new `id`, `class` and `data-object-id` attributes.

12.21 Version 0.9.2

- Fix errors in Ajax view, due to a `json` variable name conflict
- Fix support for old jQuery and new jQuery (.on vs .live)
- Fix running the example project with Django 1.5
- Fix error messages in `post_comment_ajax` view.
- Fix empty user name column in the admin list.
- Fix undesired "reply" link in the preview while using django-threadedcomments.
- Fix HTML layout of newly added threaded comments.
- Fix Python 3 support

12.22 Version 0.9.1

- Fix running at Django 1.6 alpha 1

12.23 Version 0.9

- Full support for django-threadedcomments out of the box.
- Fix CSS class for primary submit button, is now `.btn-primary`.

12.24 Version 0.8.0

First public release

- Ajax-based preview and posting of comments
- Configurable form layouts using django-crispy-forms and settings to exclude fields.
- Comment moderation, using Akismet integration and auto-closing after N days.
- E-mail notification to the site managers of new comments.
- Rudimentary support for django-threadedcomments

CHAPTER 13

Indices and tables

- genindex
- modindex
- search