PyBBM Documentation

Release rc

Pavel Zukov

Jun 04, 2017

Contents

PyBBM is a complete Django forum solution with the following features:

- Avatars
- Custom profiles and support of Custom User Model (since Django 1.5)
- Editable posts
- Pre-moderation
- Custom sanitization
- Anonymous posting
- Subscriptions
- Polls
- ...

All features are based on:

- 95% test code coverage
- Twitter Bootstrap compatible default theme
- Ready to use example projects

The main focus in the development of PyBMM is to build it in a way that allows *easy* integration in existing Django-based sites. This means that PyBMM does not provide features like user registration, password restoring or the user login page. (But the example projects show how to do these tasks ;))

Contents:

CHAPTER 1

Installation

Mandatory dependencies

PyBBM requires the following packages:

- django
- django-annoying
- unidecode (for slugifying instances for nicer urls)

By installing PyBBM with `pip` or `easy_install`, all the above dependencies will be installed automatically:

```
pip install pybbm
```

Optional dependencies

The following dependencies are optional. You can install them with `pip install`:

- For better perfomance and easy images thumbnailing you can use any thumbnail django application. PyBBM by default uses `sorl.thumbnail` if it is installed and included in your `INSTALLED_APPS` setting. It is used for defining the *avatar* field in the *PybbProfile* model and for resizing the avatar in the `pybb/avatar.html` template. If you decide to install `sorl.thumbnail` with django 1.7 you have to install at least 11.12.1b version with:

  ```
  pip install "sorl-thumbnail>=11.12.1b"
  ```

- `PIL` (Python Imaging Library) or its more up-to-date fork `Pillow` is optional if you configure `sorl.thumbnail` to use different backend or don't use `sorl.thumbnail` in general, but remember that using an ImageField in forms requires the Python Imaging Library to be installed (i.e. you should install it if you use the built-in profile).
- PyBBM emulates the behavior and functionality of `django-pure-pagination`, but we recommend to install it in your project.

- Choose from `bbcode` and `markdown` libraries if you use one of the attached to pybbm markup engines. For more information see *Markup*

Fresh project

If you start a new project based on pybbm, checkout pybbm.org website codebase from https://github.com/hovel/pybbm_org and skip the next steps.

Enable applications and edit settings

- Add the following apps to your `INSTALLED_APPS` to enable pybbm and required applications:
  - pybb

```
INSTALLED_APPS = (
    ....
    'pybb',
    ...
)
```

- Add `pybb.context_processors.processor` to your `settings.TEMPLATE_CONTEXT_PROCESSORS`:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'pybb.context_processors.processor',
    ...
    )
```

- Add `pybb.middleware.PybbMiddleware` to your `settings.MIDDLEWARE_CLASSES`:

```
MIDDLEWARE_CLASSES = (
    ...
    'pybb.middleware.PybbMiddleware',
    ...
)
```

Enable PyBBM urlconf

Put `include('pybb.urls', namespace='pybb'))` into main project `urls.py` file:

```
urlpatterns = [
    ....
    (r'^forum/', include('pybb.urls', namespace='pybb')),
    ....
]
```

Enable your site profile

Setup forum's profile model and `PYBB_PROFILE_RELATED_NAME` setting.

If you have no site profile, dafault settings will satisfy your needs.

If you have a custom user model, which stores all profile fields itself, or if you have custom site profile model, then check that it inherits from `pybb.profiles.PybbProfile` or contains all fields and properties from this class.

Then set `PYBB_PROFILE_RELATED_NAME` to `None` for custom user model, or to related_name from OneToOne field related to User from custom site profile model.

For more information see *how to use custom user model with pybbm*

Migrate database

Since django 1.7 release you have several combinations of installed packages that affect database migrations:

- **django >= 1.7** Django since 1.7 version has it's own migration engine. Pybbm fully supports django 1.7 migrations, so just run:

```
python manage.py migrate pybb
```

- We recommend to use database engine that supports transaction management (all django backends except sqlite). Otherwise you have small chance to face some inconsistency in DB after failed post/topic creation.

Templates

Check that:

- Your templates directory contains the "base.html" template. Otherwise, set a custom base template with *PYBB_TEMPLATE*.
- Basic template contains at least a `content` block.

CHAPTER 2

Settings

Basic settings

PYBB_TOPIC_PAGE_SIZE

Number of posts in topic page

Default: 10

PYBB_FORUM_PAGE_SIZE

Number of topics in forum page

Default: 10

PYBB_FREEZE_FIRST_POST

Freeze first post in topics (show on every page)

Default: False

PYBB_DEFAULT_TITLE

Default title for forum index page

Default: 'PYBB Powered Forum'

PYBB_DEFAULT_AUTOSUBSCRIBE

Users will be automatically subscribed to topic when create post in it.

Default: True

PYBB_DISABLE_SUBSCRIPTIONS

Users won't be able to subscribe to topic. If you want to have a more advanced mode than enable / disable (for example, use model permissions), you just have to overwrite the "may_subscribe_topic" method of the Permission handler. If you disabled topic subscriptions, already subscribed users will still receive notifications: see `PYBB_DISABLE_NOTIFICATIONS` to stop notifications sending.

Default: False

PYBB_DISABLE_NOTIFICATIONS

Users which have subscribed to a topic won't receive notifications but still be able to subscribe to topics. See `PYBB_DISABLE_NOTIFICATIONS` to disable topic subscription too. This is usefull if you want to to use your own notification system. Additionally, if your custom user model has a `receive_emails` field, it will be used to determine whether to send notification emails to each user.

Default: False

PYBB_USE_DJANGO_MAILER

When True and django-mailer app installed, then for sending email pybbm will use this app. With django-mailer you can manage emails from your site in queue. But in this case you have to periodically actually send emails from queue. For more information see app home page.

Default: False

Emoticons

PYBB_SMILES_PREFIX

Prefix for emoticons images set, related to STATIC_URL.

Default: 'pybb/emoticons'

PYBB_SMILES

Dict for emoticon replacement. Key - text to be replaced, value - image name.

Default:

```
{
    '&gt;_&lt;': 'angry.png',
    ':.(': 'cry.png',
    'o_O': 'eyes.png',
    '[]_[]': 'geek.png',
    '8)': 'glasses.png',
```

```
    ':D': 'lol.png',
    ':(': 'sad.png',
    ':O': 'shok.png',
    '-_-': 'shy.png',
    ':)': 'smile.png',
    ':P': 'tongue.png',
    ';)': 'wink.png'
}
```

e.g. text ";)" in post will be replaced to:

```
<img src="{{ STATIC_URL }}{{ PYBB_SMILES_PREFIX }}wink.png">
```

with default setting.

User profile settings

Next settings used only if you don't customize user profile model, user profile creation form or templates.

PYBB_PROFILE_RELATED_NAME

Related name from profile's OneToOne relationship to User model. If profile model is User model itselt then set it to *None*.

Default: 'pybb_profile'

For more information see *how to use custom user model with pybbm*

PYBB_AVATAR_WIDTH

Avatar width to use in templates (avatars scaled using sorl.thumbnail if it installed and included in project).

Default: 80

PYBB_AVATAR_HEIGHT

Avatar height to use in templates (avatars scaled using sorl.thumbnail if it installed and included in project)

Default: 80

PYBB_MAX_AVATAR_SIZE

Maximum avatar size, in bytes

Default: 51200 (50 Kb)

PYBB_DEFAULT_TIME_ZONE

Default time zone for forum as integer. E.g. setting to 1 means GMT+1 zone.

Default: 3 (Moscow)

PYBB_SIGNATURE_MAX_LENGTH

Limit of sybmols in user signature

Default: 1024

PYBB_SIGNATURE_MAX_LINES

Limit of lines in user signature

Default: 3

PYBB_DEFAULT_AVATAR_URL

Will be used if user doesn't upload avatar

Default: settings.STATIC_URL + 'pybb/img/default_avatar.jpg'

Style

You can use builtin templates with custom basic template.

PYBB_TEMPLATE

Builtin templates will inherit this template

Default: 'base.html

PYBB_TEMPLATE_MAIL_TXT

Builtin *txt* emails templates will inherit this template

Default: 'pybb/mail_templates/base.html

PYBB_TEMPLATE_MAIL_HTML

Builtin *html* emails templates will inherit this template

Default: 'pybb/mail_templates/base-html.html

Markup engines

PYBB_MARKUP

Markup engine used in forum. Also see *PYBB_MARKUP_ENGINES (deprecated)* below

Default: 'bbcode'

PYBB_MARKUP_ENGINES_PATHS

Dict with avaiable markup engines path. One of them should be selected with PYBB_MARKUP

Markup engine should be a path to a class, that inherits from *pybb.markup.base.BaseParser*. Markup engine should take care of replacing smiles in body with related emoticons.

by default PyBBM support *bbcode* and *markdown* markup:

```
{
    'bbcode': 'pybb.markup.bbcode.BBCodeParser',
    'markdown': 'pybb.markup.markdown.MarkdownParser'
}
```

Please note, that previous version of pybb used two different settings : *PYBB_MARKUP_ENGINES* and *PYBB_QUOTE_ENGINES* which were callables. This is still supported, but is deprecated.

PYBB_MARKUP_ENGINES (deprecated)

Should be the same dict with paths to markup engine classes as *PYBB_MARKUP_ENGINES_PATH* setting

Default: *PYBB_MARKUP_ENGINES_PATHS*.

For more information see *Markup*

PYBB_QUOTE_ENGINES (deprecated)

Deprecation note: Every markup class must inherit from *pybb.markup.base.BaseParser*

For more information see *Markup*

Should be the same dict with paths to markup engine classes as *PYBB_MARKUP_ENGINES_PATH* setting

Default: *PYBB_MARKUP_ENGINES_PATHS*.

Post cleaning/validation

PYBB_BODY_CLEANERS

List of paths to 'cleaner' functions for body post to automatically remove undesirable content from posts. Cleaners are user-aware, so you can disable them for some types of users.

Each function in list should accept *auth.User* instance as first argument and *string* instance as second, returned value will be sended to next function on list or saved and rendered as post body.

For example this is enabled by default *rstrip_str* cleaner:

```
def rstrip_str(user, str):
    if user.is_staff:
        return str
    return '\n'.join([s.rstrip() for s in str.splitlines()])
```

Default:

```
PYBB_BODY_CLEANERS = ['pybb.markup.base.rstrip_str', 'pybb.markup.base.filter_blanks']
```

PYBB_BODY_VALIDATOR

Extra form validation for body of post.

Called as:

```
PYBB_BODY_VALIDATOR(user, body)
```

at *clean_body* method of *PostForm* Here you can do various checks based on user stats. E.g. allow moderators to post links and don't allow others. By raising:

```
forms.ValidationError('Here Error Message')
```

You can show user what is going wrong during validation.

You can use it for example for time limit between posts, preventing URLs, etc.

Default: None

Anonymous/guest posting

PYBB_ENABLE_ANONYMOUS_POST

Allow post for not-authenticated users.

Default: False

See *anonymous posting* for details.

PYBB_ANONYMOUS_USERNAME

Username for anonymous posts. If no user with this username exists it will be created on first anonymous post.

Default: 'Anonymous'

PYBB_ANONYMOUS_VIEWS_CACHE_BUFFER

Number of anonymous views for each topic, that will be cached. For disabling caching anonymous views just set it to *None*.

Default: 100

Premoderation

PYBB_PREMODERATION

Filter for messages that require pre-moderation.

Default: False

See *Pre-moderation* for details.

Attachments

PYBB_ATTACHMENT_ENABLE

Enable attahcments for all users.

Default: False

PYBB_ATTACHMENT_SIZE_LIMIT

Maximum attachment limit (in bytes).

Default: 1048576 (1Mb)

PYBB_ATTACHMENT_UPLOAD_TO

Directory in your media path for uploaded attacments.

Default: 'pybb_upload/attachments'

Polls

Note: For disabling polls on your forum, write custom permission handler and return from *may_create_poll* method *False* See *PYBB_PERMISSION_HANDLER* setting.

PYBB_POLL_MAX_ANSWERS

Max count of answers, that user can add to topic.

Default: 10

Permissions

PYBB_AUTO_USER_PERMISSIONS

Automatically adds add post and add topic permissions to users on user.save().

Default: True

PYBB_PERMISSION_HANDLER

If you need custom permissions (for example, private forums based on application-specific user groups), you can set *PYBB_PERMISSION_HANDLER* to a class which inherits from *pybb.permissions.DefaultPermissionHandler* (default), and override any of the *filter_** and *may_** method. For details, look at the source of *pybb.permissions.DefaultPermissionHandler*. All methods from permission handler (custom or default) can be used in templates as filters, if loaded pybb_tags. In template will be loaded methods which start with 'may' or 'filter' and with three or two arguments (include 'self' argument)

Default: 'pybb.permissions.DefaultPermissionHandler'

Urls

PYBB_NICE_URL

Changes old/classics URLs to more semantic URLs using Category/Forum/Topic's slug. For example www.yourforum.com/forum/1 becomes www.yourforum.com/c/category_slug/forum_slug. Old URLs will have a permanent redirections to new ones.

Default: False

PYBB_NICE_URL_PERMANENT_REDIRECT

When PYBB is set to use PYBB_NICE_URL, this setting changes the HTTP response code used to redirect old style URL to new one. True (default) use 301 (permantent) redirect. If set to False, it uses 302 (temporary) redirect. *False* value is usefull for testing period to not loose SEO related to old URLs, then, once testing period is over, setting this to True will ensure that your old URLs will be updated to the new ones next time the Search Engine will check it.

Default: True

PYBB_NICE_URL_SLUG_DUPLICATE_LIMIT

Limit for checking duplicate slugs. After reaching this limit while trying to find unique slug ValidationError will be raised.

Default: 100

PYBB_ENABLE_ADMIN_POST_FORM

Enable admin post form that allowed staff to post with any username automagically creating it if it did not exist.

Default: True

PYBB_ALLOW_DELETE_OWN_POST

Allow non-superusers to delete their own posts.

Default: True

CHAPTER 3

How to integrate custom user model in pybbm forum

Custom user model is a great feature introduced in django framework since version 1.5. This topic describes how to integrate your custom user model in pybbm forum application.

First of all pybbm uses some fields from standard User model and permission system. The simplest way to make your custom model compatible with pybbm is to inherite from *django.contrib.auth.models.AbstractUser*

Second way is to meet next requirments:

- define USERNAME_FIELD constant, which point to unique field on your model
- define email, is_staff, is_superuser fields or properties
- inherite from *django.contib.auth.models.PermissionsMixin* or reproduce django's default permission system

Next step is to decide which model will store all fields for pybb forum profiles. This model should be referenced to current User model (custom or default) in OneToOne relationship. To easily setup such model you can use predefined *pybb.profiles.PybbProfile* class. If profile model is custom user model itself then you can use *PybbProfile* class as mixin for adding required fields. For more granunal control of fields in your forum profile model you may not rely on *PybbProfile* and define all fields from this model manually. You can use fields from another app (such as *avatar* or *language*), but you have to define proxy properties in forum profile model and build custom edit profile view, which will be override default pybb profile edit view in urls.py.

Next define *get_display_name* method for your profile model if you want custom username rendering everywhere in forum. Default *pybb.Profile* model return user's username field as display name, base *pybb.profiles.PybbProfile* class trying to return *self.user.get_username()* or *self.get_username()*.

Last step is to correctly set *PYBB_PROFILE_RELATED_NAME* setting. You have to set this setting to related_name parameter from profile's model from OneToOne relation to User model. If you use custom user model and this model is profile model itself, then you have to set this setting to *None*

CHAPTER 4

Pre-moderation

PyBBM shipped with fully customizable pre-moderation system.

Because in different circumstances you may need different pre-moderation conditions, pybbm gives you ability to create custom filter for messages that require pre-moderation.

All you need is to provide function from two arguments: *user* and *body*. This function should return *True* if message pass filter and *False* if message require pre-moderation.

For example, next filter allow to post without pre-moderation only for superusers:

```
def check_superuser(user, post):
    if user.is_superuser:
        return True
    return False
```

Told pybbm to use this function by setting *PYBB_PREMODERATION* in settings:

```
PYBB_PREMODERATION = check_superuser
```

CHAPTER 5

Permissions

Pybb allow you to manage permissions by overwriting a Permission handler. See `PYBB_PERMISSION_HANDLER` setting.

Below are the current pybb permissions managed by the default Permission handler for each type of “user”.

Permissions for anonymous:

action	can do with
PREMODERATION False	can do with PREMODERATION False
can do with PREMODERATION True	can do with PREMODERATION True
view normal forum	yes	yes
view hidden forum	no	no
view other topic	yes	yes
view other post	yes	yes
view own on moderation topic	see logged-in user	see logged-in user
view own on moderation post	see logged-in user	see logged-in user

view other on moderation topic no no

view other on moderation post yes no

add post in normal topic no no

add post in on moderation topic no no

add post in closed topic no no

edit own normal post see logged-in user see logged-in user

edit own on moderation post see logged-in user see logged-in user

edit other post no no

delete own normal post see logged-in user see logged-in user

delete own on moderation post see logged-in user see logged-in user

delete other post no no

close and unclose topic no no

stick and unstick topic no no

manage moderators no no

Permissions for a logged-in user:

action	can do with

PREMODERATION False

can do with

PREMODERATION False

can do with

PREMODERATION True

can do with

PREMODERATION True

view normal forum yes yes

view hidden forum no no

view other topic yes yes

view other post yes yes

view own on moderation topic yes yes

view own on moderation post yes yes

view other on moderation topic no no

view other on moderation post yes no

add post in normal topic yes yes

add post in on moderation topic no no

add post in closed topic no no

edit own normal post yes yes

edit own on moderation post yes yes

edit other post no no

delete own normal post yes yes

delete own on moderation post yes yes

delete other post no no

close and unclose topic no no

stick and unstick topic no no

manage moderators no no

Permissions for a moderator of the current forum:

action	can do with

PREMODERATION False

can do with

PREMODERATION False

can do with

PREMODERATION True

can do with

PREMODERATION True

view normal forum yes yes

view hidden forum no no

view other topic yes yes

view other post yes yes

view own on moderation topic see logged-in user see logged-in user

view own on moderation post see logged-in user see logged-in user

view other on moderation topic yes yes

view other on moderation post yes yes

add post in normal topic yes yes

add post in on moderation topic yes yes

add post in closed topic yes yes

edit own normal post see logged-in user see logged-in user

edit own on moderation post see logged-in user see logged-in user

edit other post yes yes

delete own normal post see logged-in user see logged-in user

delete own on moderation post see logged-in user see logged-in user

delete other post yes yes

close and unclose topic yes yes

stick and unstick topic yes yes

manage moderators no no

Permissions for a “is_staff” user without pybb permissions:

action	can do with

PREMODERATION False

can do with

PREMODERATION False

can do with

PREMODERATION True

can do with

PREMODERATION True

view normal forum	yes	yes
view hidden forum	yes	yes
view other topic	yes	yes
view other post	yes	yes
view own on moderation topic	see logged-in user	see logged-in user
view own on moderation post	see logged-in user	see logged-in user
view other on moderation topic	no	no
view other on moderation post	yes	no
add post in normal topic	yes	yes
add post in on moderation topic	no	no
add post in closed topic	no	no
edit own normal post	see logged-in user	see logged-in user
edit own on moderation post	see logged-in user	see logged-in user
edit other post	no	no
delete own normal post	see logged-in user	see logged-in user
delete own on moderation post	see logged-in user	see logged-in user
delete other post	no	no
close and unclose topic	no	no
stick and unstick topic	no	no
manage moderators	yes	yes

Permissions for a “is_staff” user with pybb permissions:

action	can do with

PREMODERATION False

can do with

PREMODERATION False

can do with

PREMODERATION True

can do with

PREMODERATION True

view normal forum	yes	yes
view hidden forum	yes	yes
view other topic	yes	yes
view other post	yes	yes
view own on moderation topic	see logged-in user	see logged-in user
view own on moderation post	see logged-in user	see logged-in user
view other on moderation topic	yes	yes
view other on moderation post	yes	yes
add post in normal topic	yes	yes
add post in on moderation topic	yes	yes
add post in closed topic	yes	yes
edit own normal post	see logged-in user	see logged-in user
edit own on moderation post	see logged-in user	see logged-in user
edit other post	yes	yes
delete own normal post	see logged-in user	see logged-in user
delete own on moderation post	see logged-in user	see logged-in user
delete other post	yes	yes
close and unclose topic	yes	yes
stick and unstick topic	yes	yes
manage moderators	yes	yes

Permissions for superuser:

action	can do with

PREMODERATION False

can do with

PREMODERATION False

can do with

PREMODERATION True

can do with

PREMODERATION True

view normal forum	yes	yes
view hidden forum	yes	yes
view other topic	yes	yes
view other post	yes	yes
view own on moderation topic	see logged-in user	see logged-in user
view own on moderation post	see logged-in user	see logged-in user
view other on moderation topic	yes	yes
view other on moderation post	yes	yes
add post in normal topic	yes	yes
add post in on moderation topic	yes	yes
add post in closed topic	yes	yes
edit own normal post	see logged-in user	see logged-in user
edit own on moderation post	see logged-in user	see logged-in user
edit other post	yes	yes
delete own normal post	see logged-in user	see logged-in user
delete own on moderation post	see logged-in user	see logged-in user
delete other post	yes	yes
close and unclose topic	yes	yes
stick and unstick topic	yes	yes

manage moderators yes yes

CHAPTER 6

Anonymous posting

PyBBM allow you to enable anonymous posting on forum.

Be very carefull by enabling anonymous posting, it is better to enable *BODY_CLEANER* setting to cleanup spam links from posts.

Enable *PYBB_ENABLE_ANONYMOUS_POST* and set *PYBB_ANONYMOUS_USERNAME* for enabling anonymous posting:

```
PYBB_ENABLE_ANONYMOUS_POST = True
PYBB_ANONYMOUS_USERNAME = 'Anonymous'
```

Carefully set *PYBB_ANONYMOUS_USERNAME*. It is better to create user with this username yourself rather than left it to autoregister on first anonymous post, somemone may want to use this username and register before first anonymous post will be posted, in that case anonymous post will share same account with this user.

Anonymous topic views count cached for each topic. The reason to do this is not update database on each anonymous request. Default value that will be cached is 100, this values controlled by *PYBB_ANONYMOUS_VIEWS_CACHE_BUFFER* setting and can be disabled by this setting too.

CHAPTER 7

Useful template tags and filters

Next filters and tags can be used when *pybb_tags* loaded in template:

- PyBBM passes all filter_* and may_* methods from current permission handler to templates as filters with **pybb_** prefix. So you can use they as:

```
{% if user|may_view_topic:topic %}
  you can view {{ topic }}
{% endif %}
```

  or:

```
{% with queryset=user|filter_topics:topic_queryset %}
  {# operations on queryset there #}
{% endwith %}
```

- *pybb_get_latest_topic* and *pybb_get_latest_posts* assigment tags you can use on every page for getting latest topics ans posts, for example for rendering in side block. Also you can pass *user* parameter (default to user from template context) for geting topics or posts available for specific user and *cnt* parameter for get specific count of topics or posts:

```
{% pybb_get_latest_topic cnt=30 as topics %}
{# operation on topics list there #}
```

- *pybb_get_profile* assigment tag can be used to get forum profile instance for any user passed as *user* argument:

```
{% pybb_get_profile user=post.user as post_user_profile %}
{# use profile fields there #}
```

CHAPTER 8

Example projects

The PyBBM source code contains two example projects under the `test/example_bootstrap` and `test/example_thirdparty` directories. Both are fully deployed and ready to use forum applications.

There is only one difference between these projects:

- `example_bootstrap` includes the LESS (a CSS preprocessor) files from Twitter Bootstrap.
- `example_thirdparty` leaves this to the thirdparty app `pinax-theme-bootstrap`.

We recommend to use `example_thirdparty` to start with. If you starting from scratch it's probaly the best way to begin.

Running the example projects

Each example directory contains requirements.txt file, you can run:

```
pip install -r requirements.txt
```

to install all dependencies.

Also *example_bootstrap/fixtures* directory contains *demo_data.json* file with some example data. So, you can run:

```
python manage.py loaddata <path to example_bootstrap>/fixtures/demo_data.json
```

from root directory to load some models in you database.

CHAPTER 9

Filtering spam, urls and other ‘bad’ stuff

PyBBM contains two settings *PYBB_BODY_VALIDATOR* and *PYBB_BODY_CLEANERS* that create very flexible system for filtering posts. Read this settings description for more information.

CHAPTER 10

Markup

How it works

Every time user save new post, message parsed by markup parser to it's html representation. It will include only allowed by engine html tags, html tags should be html-encoded and rendered as simple text to prevent XSS attacks.

Pybbm includes two default engines. Actions needed to use these engines:

- bbcode engine: install required package with `pip install bbcode` command and set *PYBB_MARKUP* to `'bbcode'`
- markdown engine: install required package with `pip install markdown` command and set *PYBB_MARKUP* to `'markdown'`

Engine classes must inherit from *pybb.markup.base.BaseParser* which defines three required methods:

- `def format(self, text)` method receives post's text as parameter and returns parsed message as html fragment
- `def quote(self, text, username='')` method receives quoted post's text and username and returns quoted string in terms of markup engine
- `def get_widget_cls(cls)` class method which return the class to use as the widget for the body field

How to change

If you want to write your custom engine you can write a new class which extends `pybb.markup.base.BaseParser`

To change behavior of one of the default parsers you can override `pybb.markup.bbcode.BBCodeParser` or `pybb.markup.markdown.MarkdownParser`.

For example, for adding additional formatter to bbcode parser you can write your own class in myproject.markup_engines.py:

```
from pybb.markup.bbcode import BBCodeParser

class CustomBBCodeParser(BBCodeParser):
    def __init__(self):
        super(CustomBBCodeParser, self).__init__()
        self._parser.add_simple_formatter('ul', '<ul>%(value)s</ul>', transform_
↪newlines=False, strip=True)
        self._parser.add_simple_formatter('li', '<li>%(value)s</li>', transform_
↪newlines=False, strip=True)
```

include it in *PYBB_MARKUP_ENGINES_PATHS* setting dict and point pybbm to use it by *PYBB_MARKUP* setting:

```
PYBB_MARKUP_ENGINES_PATHS = {'custom_bbcode': 'myproject.markup_engines.
↪CustomBBCodeParser'}
PYBB_MARKUP = 'custom_bbcode'
```

or you can override default bbcode engine in settings.py:

```
PYBB_MARKUP_ENGINES_PATHS = {'bbcode': 'myproject.markup_engines.CustomBBCodeParser'}
PYBB_MARKUP = 'bbcode' # don't required because 'bbcode' is default value
```

Using different media assets

When you define your custom markup parser you may want to control how it will be rendered. For that purpose pybbm uses django's concept of widgets and their media assets. Widget class used by markup engine controlled by `get_widget_cls` class method of engine class. By default it returns `widget_class` attribute value. Pybbm default parsers use next widgets:

- pybb.base.BaseParser - django.forms.Textarea
- pybb.bbcode.BBCodeParser - pybb.bbcode.BBCodeWidget
- pybb.bbcode.MarkdownParser - pybb.markdown.MarkdownWidget

To get it working in your templates include `{{ form.media }}` or `{{ form.media.css }}` / `{{ form.media.js }}` in proper place in every template where you use post form.

CHAPTER 11

Javascript functionality

Pybb does not depend on any javascript code. But javascript can provide more rich user experience when interacting with forum.

For enabling javascript features your installation should meet some requirements:

- include in your templates link to *pybbjs.js* file, for example with `{% static 'pybb/js/pybbjs.js' %}` tag
- to enable deleting posts via ajax, add to your delete link inline onclick handler with calling pybb_delete_post(url, post_id, confirm_text)
- to enable quoting selected text in post, add in each post link with *quote-selected-link* class
- to enable quoting full message via ajax, add in each post link with *quote-link* class and href attribute pointed to view that return text to quote
- to enable insert in post body user's nickname by clicking with shift pressed, just wrap each post with tag with *post-row* class and place inside it nickname wrapped by tag with *post-username* class

All of this features enabled in standard templates shipped with pybbm app.

CHAPTER 12

Contributing

Development happens on github, with main repo: https://github.com/hovel/pybbm

Issues, forks, patches and pull requests live here ;)

There are many ways to contribute to pybbm and you do not need to be a developper to do it:

1. Find a bug and submit an issue. You should:
   - explain what was expected
   - explain what was really done
   - explain how to reproduce (environment, steps etc.)
2. Improve documentation
   - you can correct typos or “bad english”
   - add a new documentation part (for exemple, a “how to” than could help people when using pybb in some use-case. eg: how to add news fields to *Topic* model).
3. Fix a bug. You should:
   - write a test which should fail because of this bug
   - write your bug fix
   - run all pybbm tests
   - be sure that code coverage does not decrease
   - create the pull request
4. Add a new feature:
   - quick-document it
   - open an issue and discuss about it
   - document it completely
   - add tests for all parts of this new feature

- write your new feature
- run all pybbm tests
- be sure that code coverage does not decrease
- create the pull request

CHAPTER 13

Testing

PyBBM has good unittest coverage. There is two way to test pybbm: a “multi-env way” via tox (multiple versions of Django, Python...), and a “local env way” with your version of python and locally installed python packages.

Testing with tox (recommended)

This is the recommanded way to test pybbm if you want to contribute. You must have tox installed on your system. (eg: *sudo pip install tox*)

1. Clone your github pybbm fork and go in its directory:

```
git clone git@github.com:yourGithubUsername/pybbm.git
cd pybbm
```

2. run tox:

```
tox
```

That’s all ;-). Tox will tests pybbm in multiple environnements configured in the pybbm’s tox.ini. If you want to test only a specific environnement from that list, you can run tox with the “-e” option. For exemple, this command will test pybbm only with python 2.7 and Django 1.8:

```
tox -e py27-django18
```

There is a special tox env to check code coverage called *coverage*. By running it, it will output a summary of code coverage and will generate a HTML rapport (in *htmlcov/index.html*) to see which part of code is not yet tested.

If you add new features to pybbm, ensure that lines you add are covered by tests ;-)

Testing in your local environment

If you want to contribute, you should use the "tox way" to test your contributions before creating a pull-request ! This testing way will allow you to test pybbm in your current local environnement. It is usefull if you have a specific environment which is not covered by tox.ini.

If you already have a working pybb in your environnement, you can go to the step 4. Else, steps 1-3 will allow you to have a minimal environnement to run pybb test project.

1. Your environnement must be ready to use pip and install Pillow and lxml python packages. For Debian, the simplest way is to install debian python packages with their dependencies:

```
sudo apt-get install python-pip python-lxml python-pillow
```

2. Now, your environnement is ready to install python packages via pip. Install pybbm from your github fork with the "-e" option to be able to contribute, and install the test requirements:

```
mkdir -p ~/tests/ && cd ~/tests/
pip install --user -e git+git@github.com:yourGithubUsername/pybbm.git#egg=pybbm
pip install --user -r src/pybbm/test/test_project/requirements_test.txt
```

3. Now, add your user-local pip install directory and the pybbm directory to your PYTHONPATH:

```
export PYTHONPATH=$PYTHONPATH:~/.local/lib/python2.7/site-packages/:~/tests/src/
↪pybbm
```

4. Now, you have an editable version of pybbm and you can run tests from the "test_project":

```
cd ~/tests/src/pybbm/test/test_project/
python manage.py test pybb
```

5. If you want to display a coverage summary and create a coverage HTML report:

```
pip install --user coveralls
PATH=$PATH:~/.local/bin/
cd ~/tests/src/pybbm/test/test_project/
coverage run --rcfile ../../.coveragerc manage.py test pybb
coverage report
coverage html
```

   Index HTML report is created in *htmlcov/index.html*

CHAPTER 14

PyBBM Changelog

0.18.3 -> 0.18.4

- Fix misspelling in Swedish translation

0.18.2 -> 0.18.3

- Fix 0005 and 0006 migrations (changes will not affect db, so it's ok if this migrations were already executed)

0.18.1 -> 0.18.2

- Minor fixes.

0.18 -> 0.18.1

- Minor fixes.

0.17.3 -> 0.18

- PyBBM is now compatible with Django>=1.8,<1.11
- Allow non-moderators to delete their own posts.
- Add setting to enable or disable admin post form.
- Add Swedish translation.

- Use FileField instead of ImageField when pillow is not available to make pillow depencency optional.
- Use staticfiles in all templates.
- Improve permission checking.
- Allow users to subscribe to a forum.
- Add a form to grant users moderator privileges.
- Add notification's emails HTML alternative.
- Add the ability to the user to use their attachments inside their posts to render it as link, image etc.
- Multiple fixes and improvements.

0.17 -> 0.17.3

- Fast fix for migrations for Posgres database. If you already get and applied migrations from 0.17 version (for example on MySQL DB) you can skip new migrations with *manage.py migrate pybb –fake*

0.16.1 -> 0.17

- Topic and post creation wrapped in transaction
- All topic/post/poll related forms can be overrided when custom view inherites pybbm view
- Demo data for example projects
- Using active markup engine when quoting posts via javascript
- Functionality to support disabling default pybbm subscriptions and notifications and new settings: *PYBB_DISABLE_SUBSCRIPTIONS* and *PYBB_DISABLE_NOTIFICATIONS*
- Fixed sorl.thumbnail/easy_thumbnail compatibility in standard *pybb/avatar.html* template
- Improved example projects
- Removed applying *urlize* filter over html produced by markdown parser (it doesn't play nicely with html markup as noted in django's docs)
- django 1.8 compatibility
- common django layout for test project
- use mysqlclient package for testing installation with mysql database backend on python 3
- optional enabling "nice urls" for entire forum, that looks like '<forum prefix>/c/<category slug>/<forum slug>/<topic slug>/'

0.16 -> 0.16.1

- Fast fixes

0.15.6 -> 0.16

- Django 1.7 compatibility.
- Fixed creating custom profile model of any class defined in settings with right related name to user model. *Migration note*: If you have workaround for creating profile in your code, you should remove it for preventing possible dubplicate unique key error on user creating.
- New get_display_name method for profile model used to unification displaying username through forum
- New markup processing. See *Markup*

0.15.5 -> 0.15.6

- Make all migrations compatible with custom user model. Break dependency on sorl.thumbnail in migrations
- Compatibility functions moved to compat.py module
- Email notifications optimization
- Example_bootstrap projects now based on bootstrap 3
- Fixes and improvements

0.15.4 -> 0.15.5

- Fixed bug when user can vote (or cancel vote) when topic was closed.
- Added *may_vote_in_topic* method to permission handler.
- Fixed blocking user view

0.15.3 -> 0.15.4

- Hot fixes to bbcode transform

0.15.2 -> 0.15.3

- bbcode engine simplified

0.15.1 -> 0.15.2

- Pybbm specific forms moved to views' attributes, added new functions to views to get such forms dynamically. This makes overriding pybbm forms much easier
- Moving from unmaintained postmarkup package to bbcode project as default bbcode render engine Changed output html for [code] tag. It will be <code></code> tags instead of . So you should duplicate styles applied to div.code for code html tag.

- Japanese translation

0.15 -> 0.15.1

- Hot fixes for Python 3 support
- Fixes for Chinese translation

0.14.9 -> 0.15

- Python 3 support
- Chinese translation

0.14.8 -> 0.14.9

- Two new methods added to permission handler: *may_attach_files* and *may_create_poll*. First method used for restrict attaching files to post by user. By default it depends on *PYBB_ATTACHMENT_ENABLE* setting. Second may be used to restrict some users to create/edit polls. By default it always return *True*. For disabling polls on your forum, just write custom permission handler and return from this method *False*

0.14.7 -> 0.14.8

- Improved javascript functionality: quote selected text, qoute full original message via ajax, insert nickname in post body. For enabling this functionality you should satisfy *some requirements* in your templates
- Support for nested forums
- *PybbProfile* abstract model moved to *pybb.profiles* module to avoid circular imports when checking models.

0.14.6 -> 0.14.7

- Django 1.6 compatibility
- unblock user functionality added

0.14.5 -> 0.14.6

- Cache anonymous views count for topic and save it in database only when some count reached (100 by default). This value can be changed by setting *PYBB_ANONYMOUS_VIEWS_CACHE_BUFFER*. Also added custom filter *pybbm_calc_topic_views* that calc actual views count for topic
- Fix for migration that may fails on clean mysql installation

- Fixed perfomance issue with feed views
- Using custom permissions handler in feed views

0.14.4 -> 0.14.5

- Minor fixes

0.14.3 -> 0.14.4

- Fix for migration that may fails on clean mysql installation (not fixed really, filxed after 0.14.5)
- Make example_thirdparty project bootstrap3 compatible

0.14.2 -> 0.14.3

- Show only available topics (by permission handler) in ForumView

0.14.1 -> 0.14.2

- Fixed MultipleObjectReturned when topic has more than one moderator

0.14 -> 0.14.1

- Fixed circular import issue

0.13.1 -> 0.14

- Restored views for rendering user's posts and topics and link to that views from profile info page
- Broken hard dependency from EditProfileView and EditProfileForm classes in forum
- Ability for users to cancel their poll vote
- Block user view accepts only POST requests
- If *block_and_delete_messages* passed to request.POST for block user view, then all user's messages will be deleted

0.13 -> 0.13.1

- Hotfix for rendering avatars

0.12.5 -> 0.13

- You can add first-unread get parameter to the topic url to provide link to first unread post from topic
- Removed django-mailer, pytils, sorl-thumbnail, south, django-pure-pagination from hard dependencies
- Support Custom User model introduced in django 1.5. Do not forget to define *PYBB_PROFILE_RELATED_NAME* in settings, if you don't use predefined *pybb.PybbProfile* model See *how to use custom user model with pybbm*
- Dropped support for django 1.3
- Experimental support for python 3
- Removed django-mailer from hard dependencies, you have to manually install it for using it's functionality

0.12.4 -> 0.12.5

- More flexible forms/forms fields rendering in templates Strongly recommended to check rendering of pybbm forms on your site (edit profile, poll/topic create/edit)
- Additional template for markitup preview You can override *pybb/_markitup_preview.html* to provide your styling for <code>, <pre> and other markitup tags
- Improved permissions handling see *PYBB_PERMISSION_HANDLER* setting in *settings*
- Fixed bugs and improved performance

0.12.3 -> 0.12.4

- *PYBB_USE_DJANGO_MAILER* setting

0.12.2 -> 0.12.3

- German translation

0.11 -> 0.12

- Fixed bug when the answers to poll unexpectedly deleted. Strongly recommendet to update to this version, if using polls subsystem
- Polish translation

0.10 -> 0.11

- Ability to override standard message when user doesn't login and not alowed anonymous posts by *PYBB_ENABLE_ANONYMOUS_POST* setting. It may be useful when project doesn't have `registration_register` and/or `auth_login` url names in `urls.py`
- Content in each `topic.html` and `forum.html` is wrapped in `<div>` tag with `topic` and `forum` classes accordingly

0.9 -> 0.10

- Templates are updated for 2nd version of twitter bootstrap
- Bootstrap less files removed from pybb.
- **Refactored example projects. *test* folder now contains two examples:**
  - *example_bootstrap* shows how to include bootstrap files in your project
  - *example_thirdparty* shows how to use another project like pinax-theme-bootstrap to style forum
- New poll feature added. When user creates new topic he can add poll question and some answers. Answers count can vary from 2 to *PYBB_POLL_MAX_ANSWERS* setting (10 by default)
- Dropped support for self containing CSS in pybb.css file and PYBB_ENABLE_SELF_CSS setting.

0.8 -> 0.9

The PYBB_BUTTONS setting is removed and overridable *pybb/_button_*.html* templates for *save*, *new topic* and *submit* buttons are provided in case css styling methods are not enough.

0.6 -> 0.7

If you use custom BODY_CLEANER in your settings, rename this setting to *PYBB_BODY_VALIDATOR*

0.5 -> 0.6

Version 0.6 has significant changes in template subsystem, with main goal to make them more configurable and simple.

- **CSS now not included with project.**
  - For a limited time legacy *pybb.css* can be enabled by activating PYBB_ENABLE_SELF_CSS settings (just set it for True).
- Twitter bootstrap now included in project tree
- Default templates now provide fine theme with twitter bootstrap, corresponded .less file 'pybb_bootstrap.less' and builded *pybb_bootstrap.css* can be located in static. You can find example of usage in test directory.

- **Pagination and breadcrumb templates changed:**
  - pagination template moved from *templates/pybb/pagination/* to *templates/pybb*
  - pagination template changed from plain links to ul/li list
  - breadcrumb now live in separated template and changed from plain links to ul/li list
  - *add_post_form.html* template renamed to *post_form.html*
- *PYBB_FORUM_PAGE_SIZE* default value changed from 10 to 20

CHAPTER 15

Migrating from pybb (lorien package)

PyBBM was drop-in replacement for pybb in 2010, but current replacing current version was not checked.

You can try to:

- Replace *pybb* package with *pybbm*
- Migrate forward
- Set AUTH_PROFILE
- Migrate old profiles with *manage.py migrate_profile* command

CHAPTER 16

PyBB Modified (PyBBM) and original PyBB

PyBB originally developed by lorien in 2008-2010 has ben dropped from development in mid 2010.

This is a list of differences between PyBB and PyBBM as of mid 2011.

- All settings of pybbm have default values, see default.py file for detailed list.
- pybbm templates fill *content*, *head*, *title* and *breadcrumb* blocks for template defined in settings as *PYBB_TEMPLATE* ("base.html" by default).
- Markup engines can be configured as an ordinary settings.
- PyBBM designed to fit django-staticfiles (django <= 1.2) or django.contrib.staticfiles (django >= 1.3).
- Default pybbm templates and css files include only layout, minimal design and no coloring, so it's easy to fit any existed site colorscheme.
- PyBBM code covered with tests more than 80%
- PyBBM provides completely rewritten unread tracking with big performance improvement on large database
- Views rewritten to use as many generics as possible
- Number of external dependencies significantly reduced
- PyBBM well documented
- PyBBM included two example projects for fast start.

CHAPTER 17

Known problems

Using with a database that doest not support microseconds

If you are using a database which does not support microseconds (MySQL before v5.7 for eg.), a forum can be wrongly marked read. It can happens if a user who read the only unread topic from a forum in the same time an other user create / update an other topic on the same forum.

CHAPTER 18

Indices and tables

- genindex
- modindex
- search