django translation manager Documentation
Release 0.3.2

COEX

Mar 07, 2018

Contents

This is the documentation for Translation Manager 0.3.2, generated Mar 07, 2018.

Contents:

CHAPTER 1

Installation

1.1 Download / Installation

The easiest way to get and install Django Translation Manager is via pip, feel free to omit the first two lines if you don't want to use virtualenv

```
virtualenv env --no-site-packages
source env/bin/activate
pip install django-translation-manager
```

In case you are curious about the source, willing to contribute, or you just want to do it yourself, feel free to see our GitHub project page

After you have installed the package, it's time for configuration

1.2 Configuratuion

1. First, add translation_manager to INSTALLED_APPS to your project's settings in *settings.py*.

   We're calling our project *project* for example's sake

   ```
   INSTALLED_APPS = (
       'django.contrib.admin',
       'django.contrib.auth',
       # ...
       # this is what we have added:
       'translation_manager',
   )
   ```

2. Next, add the following variables to your settings and set them accordingly

```python
# Required paths to all locale dirs
LOCALE_PATHS = []

# Path to project basedir / workdir - root folder of project
# TRANSLATIONS_BASE_DIR = os.path.dirname(os.path.dirname(__file__))
TRANSLATIONS_BASE_DIR = ''

# Language to display in hint column to help translators
# see translation of string in another language
TRANSLATIONS_HINT_LANGUAGE = ''
```

3. add post_save signal to restart the server:

```python
from translation_manager.signals import post_save as translation_post_save

translation_post_save.connect(restart_server, sender=None)
```

4. use syncdb or migrate

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

5. Now load strings from .po files via python shell

```
./manage.py shell
```

```python
from translation_manager.manager import Manager

m = Manager()
m.load_data_from_po()
```

6. if you need, add a link to translation admin

```
{% url admin:translation_manager_translationentry_changelist %}
```

You should now have your Django Translation Manager up and running

Configuration

You can configure Django Translation Manager by overriding it's default settings in your settings.py Default settings can be found in defaults.py in django translation manager folder.

2.1 Mandatory settings

Those have to be explicitly set for Django Translation Manager to work

This list should contain every directory containing locale directories

```
# Required paths to all locale dirs
# LOCALE_PATHS = [
#     '/foo/bar/locale',
#     '/foo/foo/bar/locale',
# ]
LOCALE_PATHS = []
```

typically the same as project base directory

```
# Path to project basedir / workdir - root folder of project
# TRANSLATIONS_BASE_DIR = '/foo/bar'
TRANSLATIONS_BASE_DIR = ''
```

Language to display in hint column to help translators see translation of string in another language

```
# Language to display in hint column to help translators
# see translation of string in another language
# TRANSLATIONS_HINT_LANGUAGE = 'foo'
TRANSLATIONS_HINT_LANGUAGE = ''
```

2.2 Optional settings

```
# Mode the translation manager behaves when creating
# translations mainly from multiple locale files
# Default value is N, where translation in DB are
# only once for specific locale file.
# Another option is P, where is translation manager
# promiscuous and creates for every translation it's
# instance for every locale file. It's useful, i.e.
# if you want has original system translations and
# also client's custom translations

TRANSLATIONS_MODE = "N"
```

```
# For storing all translations to db regardless they have
# any occurrences or not set True, otherwise set False.
# If False only translations having occurrences in your
# application will be stored.
TRANSLATIONS_ALLOW_NO_OCCURRENCES = False
```

```
# Dirs and files ignored for makemessages.
# TRANSLATIONS_IGNORED_PATHS = ['env', 'foo', 'bar']
TRANSLATIONS_IGNORED_PATHS = ['env']
```

```
# Backup on make messages:
TRANSLATIONS_MAKE_BACKUPS = True
```

```
# Clean .po files (delete content) after backup (this prevents duplicities)
TRANSLATIONS_CLEAN_PO_AFTER_BACKUP = True
```

```
# Forced filters on changelist queryset.
# Uses ORed original__contains Django ORM filter.
# TRANSLATIONS_QUERYSET_FORCE_FILTERS = ['foo', 'bar']
TRANSLATIONS_QUERYSET_FORCE_FILTERS = []
```

```
# Relative path to locale dir with hint languages
# Current locale path of translated string used by default
TRANSLATIONS_HINT_LANGUAGE_FORCED_RELATIVE_LOCALE_PATH = ''
```

```
# exclude fields from administration:
TRANSLATIONS_ADMIN_EXCLUDE_FIELDS = []
```

```
# define admin fields manually: for all fields look to admin.py:default_fields
TRANSLATIONS_ADMIN_FIELDS = []
```

```
# tuple of title and list of regex expression used for filtering in administration.
# Each object should be a tuple of (regex_filter, label)
TRANSLATIONS_CUSTOM_FILTERS = []
```

CHAPTER 3

Contributing to Django Translation Manager

3.1 How to Contribute

Django Translation Manager is now on GitHub, so all contributions should be either associated with a pull request or with a ticket & patch.

3.2 Contribution Guidelines

Code submitted should:

- Be formatted according to the PEP8 style guideline **except** that it does not need to confirm to the 79-column limit requirement of the guideline.

- Use naming schemes consistent with Django Translation Manager conventions

License

4.1 Mozilla Public License 2.0

http://choosealicense.com/licenses/mpl-2.0/

CHAPTER 5

Donate

Thank you!

We really appreciate every donation that helps us take Django Translation Manager to the next level.

Indices and tables:

- genindex
- modindex
- search