django-version-manager Documentation

Release 0.1.0

Eray Erdin

Apr 24, 2019

Contents

CHAPTER 1

Models

Django Version Manager has custom models to enhance your version management. However, you need to know that these models are application-unaware, which means they do not really know about your application's name (machine-readable or human-readable).

1.1 Version

Version is a *model* containing the following information about your version:

Field	Field Type	Nullable	Default	Description
major	PositiveIntegerField	False	-	Major version of your application.
minor	PositiveIntegerField	False	-	Minor version of your application.
patch	PositiveIntegerField	False	-	Patch version of your application.
status	CharField(16)	False	-	Status of your application.
release_date	DateTimeField	False	-	Release date of your application.

The field *status* can only take the values below:

- prealpha
- alpha
- beta
- stable

Instead of memorizing these, you can use *django_version_manager.status.Status* class. *Status* class extends *enum.Enum*, which means you have to extract the value. See a *Version* object creation example as following:

```
Version.objects.create(
    ...
    status=Status.PREALPHA.value
)
```

The standard querysets based on *Version* model is ordered due to their *release_date* property in a *reversed* manner. So, if you use *filter* or *all* queries on *Version* model, the first instance will have *the latest release date*.

1.2 Changelog

Changelog helps you write metadata about a version of your application. It is directly linked to a *Version* instance in a one-to-one relationship.

Field	Field Type	Nullable	Default	Description
version	OneToOneField	False	-	Version instance.
log	TextField	False	-	A log of your version.

CHAPTER 2

Templating

Django Version Manager has a couple of helper template tags to expose your application's version info. However, before using any of template tags, you need to load it by adding the snippet below to the top of your templates:

```
{% load version_manager %}
```

2.1 latest_version

You can use *latest_version* tag to print out latest version of your application.

```
{% latest_version %}
```

CHAPTER 3

Contributing

Contributing is very much welcome.

3.1 Pull Requests

If you want to send PR, you need to follow some conventions to contribute to this repository:

- You need to fork the code and create a new branch.
- Branch naming convention is followed by phord's answer on Stackoverflow.

3.2 Issues

Any kind of issues are also welcome in this repository. Since the project is quite new, no conventions were applied to the issues. However, it would be great if you followed standard issue convention, covering the answers to the questions below:

- What did you expect?
- What happened?
- What is the environment? (Including this project's version.)

CHAPTER 4

Introduction

Django Version Manager is a Django library in case you want to expose your version and metadata about it to your users.

Why would you want this anyway? In case you are doing a progressive development on your project, you may like to inform or notify users about your version and latest changes that has happened.

CHAPTER 5

Requirements and Support

This library is already tested on these environments:

- Python 3.5
- Django 1.9 and onwards

This does not mean it will not work out of bounds, but you need to clone the code and run *nose* and *tox* tests yourself. It would be kind if you sent PR of your test or opened an issue about your test results.

CHAPTER 6

Installing

To install this package, use *pip* as follows:

```
pip install django-version-manager
```

Then, you need to include *Django Version Manager* to your *INSTALLED_APPS* as follows:

```
INSTALLED_APPS = [
    ...
    'django_version_manager'
]
```

Check out other sections to learn more about this library.