django-project-portfolio Documentation

Release 1.3.1

James Bennett

Jun 07, 2017

Contents

1 Documentation contents **3**

Python Module Index **9**

django-project-portfolio is a Django application for displaying information about software projects you maintain.

Models are included for storing information about projects, versions of projects, the status of each release and associated metadata including licensing and location of packages, source code, documentation and continuous integration. *Built-in views* are provided for basic display of this information.

CHAPTER 1

Documentation contents

Installation guide

The 1.3 release of django-project-portfolio supports Django 1.8, 1.10, and 1.11 on the following Python versions (matching the versions supported by Django itself):

- Django 1.8 supports Python 2.7, 3.3, 3.4, and 3.5.
- Django 1.10 supports Python 2.7, 3.4, and 3.5.
- Django 1.11 supports Python 2.7, 3.4, 3.5, and 3.6

Important: Python 3.2

Although Django 1.8 supported Python 3.2 at the time of its release, the Python 3.2 series has reached end-of-life, and as a result support for Python 3.2 has been dropped from django-project-portfolio.

If you do not already have a supported version of Django installed, installing django-project-portfolio will install the latest release of Django automatically as a dependency.

Normal installation

The preferred method of installing django-project-portfolio is via `pip`, the standard Python package-installation tool. If you don't have `pip`, instructions are available for how to obtain and install it. If you're using Python 2.7.9 or later (for Python 2) or Python 3.4 or later (for Python 3), `pip` came bundled with your installation of Python.

Once you have `pip`, type:

```
pip install django-project-portfolio
```

Installing from a source checkout

If you want to work on django-project-portfolio, you can obtain a source checkout.

The development repository for django-project-portfolio is at <https://github.com/ubernostrum/django-project-portfolio>. If you have git installed, you can obtain a copy of the repository by typing:

```
git clone https://github.com/ubernostrum/django-project-portfolio.git
```

From there, you can use normal git commands to check out the specific revision you want, and install it using `pip install -e .` (the `-e` flag specifies an "editable" install, allowing you to change code as you work on django-project-portfolio, and have your changes picked up automatically).

Configuration and use

Once you have Django and django-project-portfolio installed, check out the quick start guide to see how to get your contact form up and running.

Models for software projects

django-project-portfolio provides three models which work together to describe software projects: *`Project`* represents a software project, *`Version`* represents a particular version of a project, and *`License`* represents the license under which a particular version is released.

class `projects.models.`**`License`**

The license under which a particular *`Version`* is released. This is tied to `Version` rather than *`Project`* in order to allow the possibility of relicensing from one version to another.

A `License` has three fields, all of which are required:

`name`

`CharField(max_length=255)`

The name of the license (for example, "GPLv2" or "MIT").

`slug`

`SlugField` (prepopulated from *`name`*)

A short, descriptive URL-safe string to identify the license. Currently there are no views in django-project-portfolio which make use of this, but the field is provided so that custom views can make use of it.

`link`

`URLField`

A link to an online version of the license's terms, or to a description of the license. For open-source licenses, individual license pages in the OSI license list are useful values for this field.

class `projects.models.`**`Project`**

A software project.

Four fields (all required) provide basic metadata about the project:

`name`

`CharField(max_length=255)`

The name of the project.

slug
`SlugField` (prepopulated from *name*)

A short, descriptive URL-safe string to identify the project.

description
`TextField`

A free-form text description of the project.

status
`IntegerField` with choices

Indicates whether the project is public or not. May be expanded to include additional options in future versions, hence the implementation as an `IntegerField` with choices instead of a `BooleanField`. Valid choices are:

PUBLIC_STATUS
Indicates a project which is public; this will cause built-in views to list and display the project.

HIDDEN_STATUS
Indicates a project which is hidden; built-in views will not list or display the project.

Four additional fields, all optional, allow additional useful data about the project to be specified:

package_link
`URLField`

URL of a location where packages for this project can be found.

repository_link
`URLField`

URL of the project's source-code repository.

documentation_link
`URLField`

URL of the project's online documentation.

tests_link
`URLField`

URL of the project's online testing/continuous integration status.

One utility method is also defined on instances of `Project`:

latest_version()
Returns the latest *Version* of this project (as defined by the `is_latest` field on `Version`), or `None` if no such version exists.

Finally, the default manager for `Project` defines one custom query method, `public()`, which returns only instances whose *status* is *PUBLIC_STATUS*. This is implemented via a custom `QuerySet` subclass, so the method will be available on any `QuerySet` obtained from `Project` as well.

class projects.models.**Version**
A particular version of a software project.

There are six fields, all of which are required:

project
`ForeignKey` to *Project*

The project this version corresponds to.

version
`CharField(max_length=255)`

A string representing the version's identifier. This is deliberately freeform to support different types of versioning systems, but be aware that it will (with the built-in views) be used in URLs, so URL-safe strings are encouraged here.

is_latest
`BooleanField`

Indicates whether this is the latest version of the project. When a `Version` is saved with `is_latest=True`, a `post_save` signal handler will toggle all other versions of that *`Project`* to `is_latest=False`.

status
`IntegerField` with choices

The status of this version. Valid choices are (taken from the Python Package Index's status choices):

PLANNING_STATUS
This is an early/planning version.

PRE_ALPHA_STATUS
This is a pre-alpha version.

ALPHA_STATUS
This is an alpha version.

BETA_STATUS
This is a beta version.

STABLE_STATUS
This is a stable version.

license
`ForeignKey` to *`License`*

The license under which this version is released.

release_date
The date on which this version was released.

Additionally, the default manager for `Version` defines one custom query method, `stable()`, which returns only instances whose *`status`* is *`STABLE_STATUS`*. This is implemented via a custom `QuerySet` subclass, so the method will be available on any `QuerySet` obtained from `Version` as well, and also on any related `QuerySet` obtained through an instance of *`Project`*.

Views for software projects

django-project-portfolio provides four built-in views for displaying information about software projects. Though not all possible views of the data are included here, the built-in views strive to cover the common cases.

class `projects.views.`**`ProjectDetail`**
Subclass of Django's generic DetailView.

Detail view of a *`Project`*. Has one required argument which must be captured in the URL:

`slug`

The *`slug`* of the project.

By default, this view will only display projects whose *`status`* is *`PUBLIC_STATUS`*.

class `projects.views.`**`ProjectList`**

Subclass of Django's generic ListView.

List of *`Project`* instances.

By default, this view will only display projects whose *`status`* is *`PUBLIC_STATUS`*.

class `projects.views.`**`VersionDetail`**

Subclass of Django's generic DetailView.

Detail view of a *`Version`*. Has two required arguments which must be captured in the URL:

`project_slug`

The *`slug`* of the *`Project`* with which this `Version` is associated.

`slug`

The *`version`* of the `Version`.

By default, only versions associated with a *`Project`* whose *`status`* is *`PUBLIC_STATUS`* can be displayed.

class `projects.views.`**`LatestVersionList`**

Subclass of `django.views.generic.ListView`.

List of the latest *`Version`* of each public (i.e., *`status`* is *`PUBLIC_STATUS`*) *`Project`*.

Python Module Index

p

projects.models, 4
projects.views, 6

Index

A

ALPHA_STATUS (projects.models.Version attribute), 6

B

BETA_STATUS (projects.models.Version attribute), 6

D

description (projects.models.Project attribute), 5
documentation_link (projects.models.Project attribute), 5

H

HIDDEN_STATUS (projects.models.Project attribute), 5

I

is_latest (projects.models.Version attribute), 6

L

latest_version() (projects.models.Project method), 5
LatestVersionList (class in projects.views), 7
License (class in projects.models), 4
license (projects.models.Version attribute), 6
link (projects.models.License attribute), 4

N

name (projects.models.License attribute), 4
name (projects.models.Project attribute), 4

P

package_link (projects.models.Project attribute), 5
PLANNING_STATUS (projects.models.Version attribute), 6
PRE_ALPHA_STATUS (projects.models.Version attribute), 6
Project (class in projects.models), 4
project (projects.models.Version attribute), 5
ProjectDetail (class in projects.views), 6
ProjectList (class in projects.views), 6
projects.models (module), 4
projects.views (module), 6
PUBLIC_STATUS (projects.models.Project attribute), 5

R

release_date (projects.models.Version attribute), 6
repository_link (projects.models.Project attribute), 5

S

slug (projects.models.License attribute), 4
slug (projects.models.Project attribute), 4
STABLE_STATUS (projects.models.Version attribute), 6
status (projects.models.Project attribute), 5
status (projects.models.Version attribute), 6

T

tests_link (projects.models.Project attribute), 5

V

Version (class in projects.models), 5
version (projects.models.Version attribute), 5
VersionDetail (class in projects.views), 7