django-autoconfig Documentation

Release 0.7.3

Mike Bryant

Sep 27, 2017

Contents

Automatic configuration of a Django project based on the requirements of apps in the INSTALLED_APPS setting.

CHAPTER 1

Basic Usage

Import `django_autoconfig.autoconfig` in `settings.py`, and call `configure_settings` with `globals()`:

```
from django_autoconfig.autoconfig import configure_settings
configure_settings(globals())
```

`django-autoconfig` will run through each app in `INSTALLED_APPS`, applying the configuration in their `autoconfig` module.

Note: `configure_settings` must be run after `INSTALLED_APPS` is defined.

In your app, define a `autoconfig` module, that contains the settings you need defined, or the app's requirements:

```
SETTINGS = {
    'MY_APP_MUST_HAVE_THIS_VARIABLE_SET': False,
}
```

CHAPTER 2

Ordering Relationships

If your app requires a particular ordering of the values in a setting, you can define a list of `django_autoconfig.autoconfig.OrderingRelationship` objects specifying these relationships.

class `django_autoconfig.autoconfig.`**`OrderingRelationship`**`(`*setting_name*, *setting_value*, *before=None*, *after=None*, *add_missing=True*`)`

Bases: `object`

This class defines a relationship between an element in a setting that's a list and one or more other entries.

It's intended to be used in an autoconfig.py file like so:

```
RELATIONSHIPS = [
    OrderingRelationship(
        'INSTALLED_APPS',
        'my.app',
        before = [
            'django.contrib.admin',
        ],
        after = [
        ],
    )
]
```

CHAPTER 3

Autoconfig Rules

1. If a setting does not exist, it will be defined.
2. If a setting exists and is a `list` or `tuple`, the contents will be appended to the existing setting, ignoring any duplicates.
3. If a setting exists and is a `dict`, the keys will be merged, and values merged, according to these same rules.
4. If an app is in `AUTOCONFIG_DISABLED_APPS`, that app won't have its autoconfig processed.

CHAPTER 4

Autoconfig urlconf

To aid in URL configuration, an automatic urlconf is provided. This can be used as follows:

```
ROOT_URLCONF = 'django_autoconfig.autourlconf'
```

This will result in each application being included under it's import path, e.g. `INSTALLED_APPS = ['app']` will result in `/app/` being mapped to `app.urls`

In addition you may define `AUTOCONFIG_INDEX_VIEW` in your settings file, this may be anything that can be passed to `reverse()`. This will create a redirect at the top of the url conf (`/`)

If you don't want a particular app to be included in the automatic urlconf, you can include the setting `AUTOCONFIG_URLCONF_EXCLUDE_APPS`, which should be a list of app names that should not be included. These apps will be skipped when the automatic urlconf is generated.

CHAPTER 5

Inconsistent States

If autoconfig cannot reach a consistent state, an `ImproperlyConfigured` exception will be raised. This means that two or more apps could not agree on the required settings, and this must be manually resolved.

Index