django-reusable-app-docs Documentation

Release 0.1.0

Brian Rosner

December 13, 2016

Contents

1 Coding Conventions **3**

2 Django Conventions **5**

2.1 How do I distribute my app? 5

2.2 Django Reusable App Conventions 5

2.3 Django Project Conventions 8

3 Indices and tables **11**

A reusable Django app, is an app that is easily plugged into a project, providing a very specific piece of functionality. They should be focused and follow the Unix philosophy of "Do one thing and do it well." Please refer to James Bennett's Djangocon talk on the subject for more info.

CHAPTER 1

Coding Conventions

- All code should follow PEP8 as closely as reasonable.
- All code should preferably follow Django's coding conventions as well.

CHAPTER 2

Django Conventions

2.1 How do I distribute my app?

Django should be using the standard Python Package Index aka Pypi or the Cheese Shop. I wrote a tutorial about how to easily package and upload your app to Pypi. All reusable apps should be uploading to Pypi.

If you upload your app to Pypi, it is generally a good idea to prefix your project name with "django-"

Also note, that when below when we refer to the default place for something as a file, that also means that you can make a directory of that same name; as per normal python.

2.2 Django Reusable App Conventions

Contents:

2.2.1 Admin

- Not required
- Placed in APP/admin.py
- Admin classes for a MODEL are called MODELAdmin

2.2.2 Context Processors

- Placed in APP/context_processors.py

2.2.3 Documentation

- Placed in a docs directory at the same level as the APP directory (you do have a top-level directory above your app, right?)
- Can contain templates, for reference

2.2.4 Feeds

- Placed in APP/feeds.py

2.2.5 Forms

- Placed in APP/forms.py

2.2.6 Managers

- Placed in APP/managers.py

2.2.7 Middleware

- Placed in APP/middleware.py
- As minimal as possible

2.2.8 Models

- Placed in APP/models (.py or a directory)
- Follow Django's model conventions

2.2.9 Templates

- Placed in APP/templates/APP/template.html

In an effort to standardize the block names in Django templates, I have proposed the following blocks for common usage.

- {% block title %}

This will be the block where you define the title of the page. Presumably your base.html will define your Site's name (perhaps even using the Sites framework) outside of this tag, to be places on all pages.

- {% block extra_head %}

I think that this is a good one that most people are already using in some form or another. In your base template you have a set of things in your ' ' that every page will have. However, a lot of other pages need things that they also want to put in the head of a document, like RSS feeds, Javascript, CSS, and all the other things that should go in the head. You can and probably will have other specialized blocks (like title above) that will fill in other parts of the head too.

- {% block body %}

This tag will be placed around the entire body section of the page. This allows you to create pages in your app that replace the entire page, not just the content. This won't be used a lot, but it's a really handy tag to have when you need it. If you haven't noticed, I've been trying to keep tag names consistent with their html tag names whenever possible.

- {% block menu %}

This would be where your menu goes. It would be the site-wide navigation, and not a per-page type of navigation.

- {% block content %}

This will be the place where you define the content on a page. This will presumably change on every page. It will not include any site navigation, headers, footers, or anything else that would belong in a base template.

Other possible blocks

- {% block content_title %}

This would be where the "title" of a content block would be. It includes the title of a blog. It can also include some kind of navigation between content, or other things like that. Presumably something that isn't the main pages content. I don't know if this should go inside the *content* tag and have a *main_content* as opposed to the *content* tag proposed above.

- {% block header %} {% block footer %}

Any text area in the header of footer that might change on a page-by- page basis.

- {% block body_id %} {% block body_class %}

This would be used to set the class or id of the body tag in the document. This is useful to set for styling and other properties.

- {% block [section]_menu %} {% block page_menu %}

This would be opposed to the *menu* block proposed above. It would be for the section or page.

2.2.10 Template Tags

- Placed in APP/templatetags/APP_tags.py

Proposed template tag syntax

- as (Context Var): This is used to set a variable in the context of the page
- for (object or app.model): This is used to designate an object for an action to be taken on.
- limit (num): This is used to limit a result to a certain number of results.
- exclude (object or pk): The same as for, but is used to exclude things of that type.

2.2.11 Tests

- Placed in APP/tests (.py or a directory)
- Fixtures are placed in APP/fixtures/fixture.json
- Usually overrides Django's testcase

2.2.12 Urls

- Placed in APP/urls (.py or a directory)
- Should have a name attribute so they are reversible; APP_MODEL_VIEW, for example blog_post_detail or blog_post_list.

2.2.13 Views

- Placed in APP/views (.py or a directory)
- Can be any callable python function.
- These should provide sane defaults, easily overridable:

Example:

```
`   def register(request, success_url=None,
              form_class=RegistrationForm
              template_name='registration/registration_form.html',
              extra_context=None):
`
```

2.3 Django Project Conventions

Contents:

2.3.1 What is a Django Project

A project in Django is a simple structure that contains a settings file, urls, and a collection of Django Apps. These can be things you have written yourself, or something third party that you choose to include in your project.

2.3.2 Suggested Project layout

```
example.com/
   settings.py
   urls.py
   local_apps/
       custom apps written for this project (Preferably Reusable; Probably on your PYTHONPATH)
   external_apps/
        External reusable apps you are using on this project
        Note: These apps can also live anywhere on your PYTHONPATH
   projects/
       dev_example/
       production_example/
       django96_example/
           manage.py, settings, urls, etc
    docs/
       This will hold the documentation for your project
   static/
       -In production this will be the root of your MEDIA_URL
       css/
       js/
       images/
   tests/
       - Project level tests (Each app should also have tests)
   uploads/
       - content imgs, etc
   templates/
       - This area is used to override templates from your reusable apps
       flatpages/
```

```
    comments/
    example/
    app1/
    app2/
```

2.3.3 Settings

- Placed in PROJECT/settings.py
- If you are using an import mechanism, please use *from local_settings import ** at the bottom of your settings.py file

2.3.4 URLs

- Placed at PROJECT/urls.py
- Should contain as little logic as possible, mostly just a pointer to each of your apps specific URLConf's.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

C

Coding Conventions, 1
    Django, 1
    Python, 1

D

Django
    Coding Conventions, 1
Djangocon Talk
    Reusable Apps, James Bennett, 1

J

James Bennett
    Djangocon Talk Reusable Apps, 1

P

PEP8, 1
Python
    Coding Conventions, 1

R

Reusable Apps
    James Bennett Djangocon Talk, 1
    What are, 1

W

What are
    Reusable Apps, 1