Django notes

# Things to remember

Static pages—use flatpages app: <https://docs.djangoproject.com/en/1.8/ref/contrib/flatpages/>

URL patterns: <https://docs.djangoproject.com/en/1.8/releases/1.8/#django-conf-urls-patterns>

# Code snippets

Add a class to the active menu item:

<ul class = 'nav'>

<li class="{% ifequal request.path 'about/'%} active {% endifequal%}">

<a href="{% url "about" %}">About</a>

</li>

</ul>

Timestamp model:

# core/models.py

from django.db import models

class TimeStampedModel(models.Model):

"""

An abstract base class model that provides selfupdating

``created`` and ``modified`` fields.

"""

created = models.DateTimeField(auto\_now\_add=True)

modified = models.DateTimeField(auto\_now=True)

class Meta:

abstract = True

# flavors/models.py

from django.db import models

from core.models import TimeStampedModel

class Flavor(TimeStampedModel):

title = models.CharField(max\_length=200)

URL namespaces:

# urls.py at root of project

urlpatterns += [

url(r'^tastings/', include(‘tastings.urls’,

namespace='tastings')),

]

# tastings/views.py snippet

class TasteUpdateView(UpdateView):

model = Tasting

def get\_success\_url(self):

return reverse("tastings:detail",

kwargs={"pk": self.object.pk})

# in template

{% extends "base.html" %}

{% block title %}Tastings{% endblock title %}

{% block content %}

<ul>

{% for taste in tastings %}

<li>

<a href="{% url "tastings:detail" taste.pk %}">{{ taste.title }} </a>

<small>

(<a href="{% url "tastings:update" taste.pk %}">update</a>)

</small>

</li>

{% endfor %}

</ul>

{% endblock content %}

Simplest views:

# simplest\_views.py

from django.http import HttpResponse

from django.views.generic import View

# The simplest FBV

def simplest\_view(request):

# Business logic goes here

return HttpResponse("FBV")

# The simplest CBV

class SimplestView(View):

def get(self, request, \*args, \*\*kwargs):

# Business logic goes here

return HttpResponse("CBV")

# *Two Scoops* Notes

Keep code readable:

* Avoid abbreviating variable names.
* Write out your function argument names.
* Document your classes and methods.
* Comment your code.
* Refactor repeated lines of code into reusable functions or methods.
* Keep functions and methods short. A good rule of thumb is that scrolling should not be necessary to read an entire function or method.

Project layout examples:

<repository\_root>/

<django\_project\_root>/

<configuration\_root>/

icecreamratings\_project/

.gitignore

Makefile

docs/

README.rst

requirements.txt

icecreamratings/

manage.py

media/ # Development ONLY!

products/

profiles/

ratings/

static/

templates/

config/

\_\_init\_\_.py

settings/

urls.py

wsgi.py

Use Cookiecutter to generate your boilerplate from cookiecutter-django.

Check chapter 6 for list of when not to use null=True.

Use get\_object\_or\_404() in views when retrieving a single item.

Guidelines When Working With CBVs

ä Less view code is better.

ä Never repeat code in views.

ä Views should handle presentation logic. Try to keep business logic in models when possible,

or in forms if you must.

ä Keep your views simple.

ä Don’t use CBVs to write custom 403, 404, and 500 error handlers. Use FBVs instead.

ä Keep your mixins simpler.

# Behavior

* User views product page
* User decides to add product to order
  + User chooses a quantity and hits submit
* User is redirected to a page of related items
  + User can add all related items to order from this page
    - Item that has been added is indicated by a color change or message
* User is redirected to a form to fill out her contact information and order information
* User receives email with order information
* Store receives email with order information
* After user is contacted, order is marked complete/contacted

# Packages to use

django-model-utils: <https://django-model-utils.readthedocs.org/en/latest/>

For timestamps

django-vanilla-views; <http://django-vanilla-views.org/>

For easy CBV

django-braces: <https://django-braces.readthedocs.org/en/latest/>

For permissions and form mixins