A Django Walkthrough

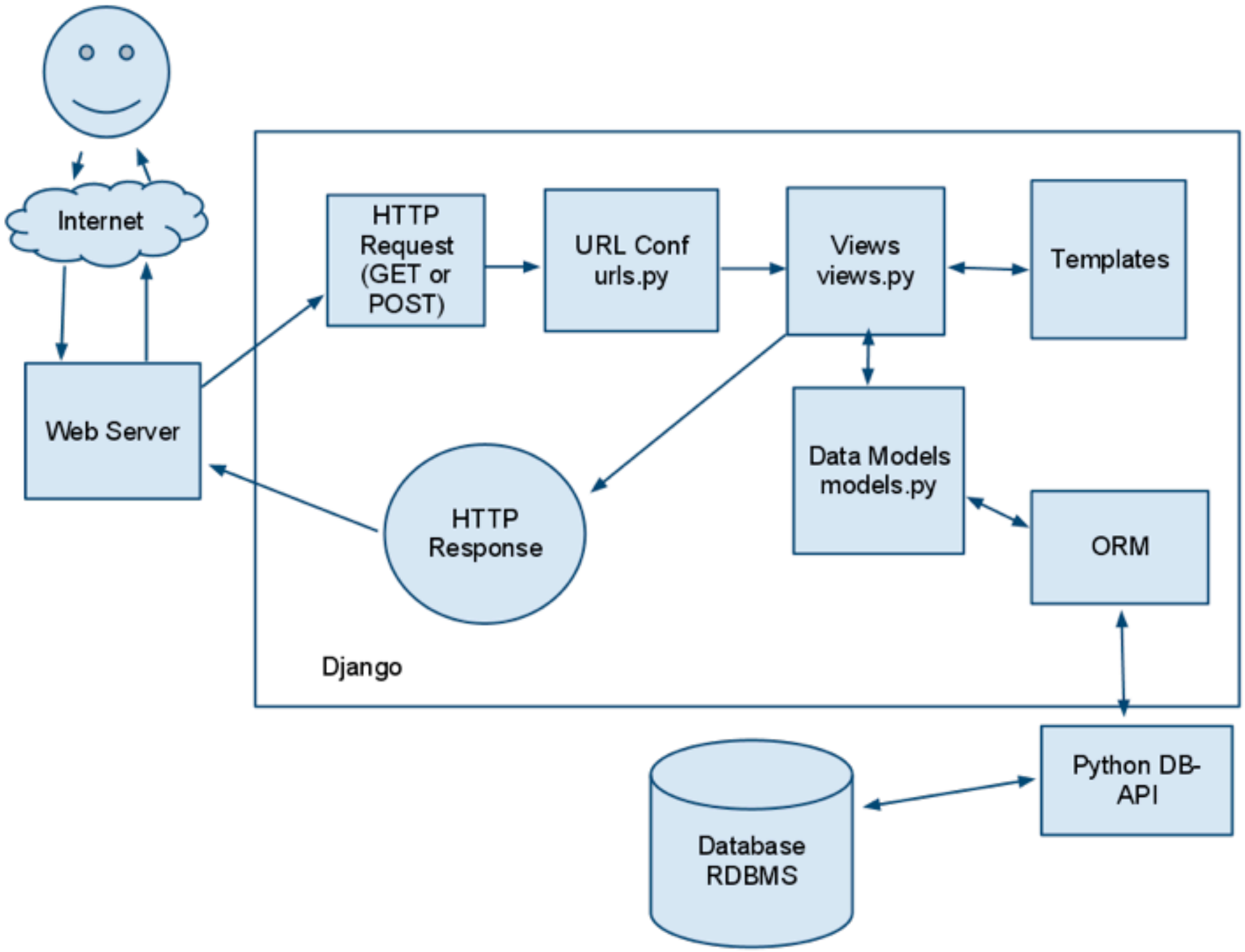
Memphis Python User Group; Jan 16, 2011

Brad Montgomery (@bkmontgomery)

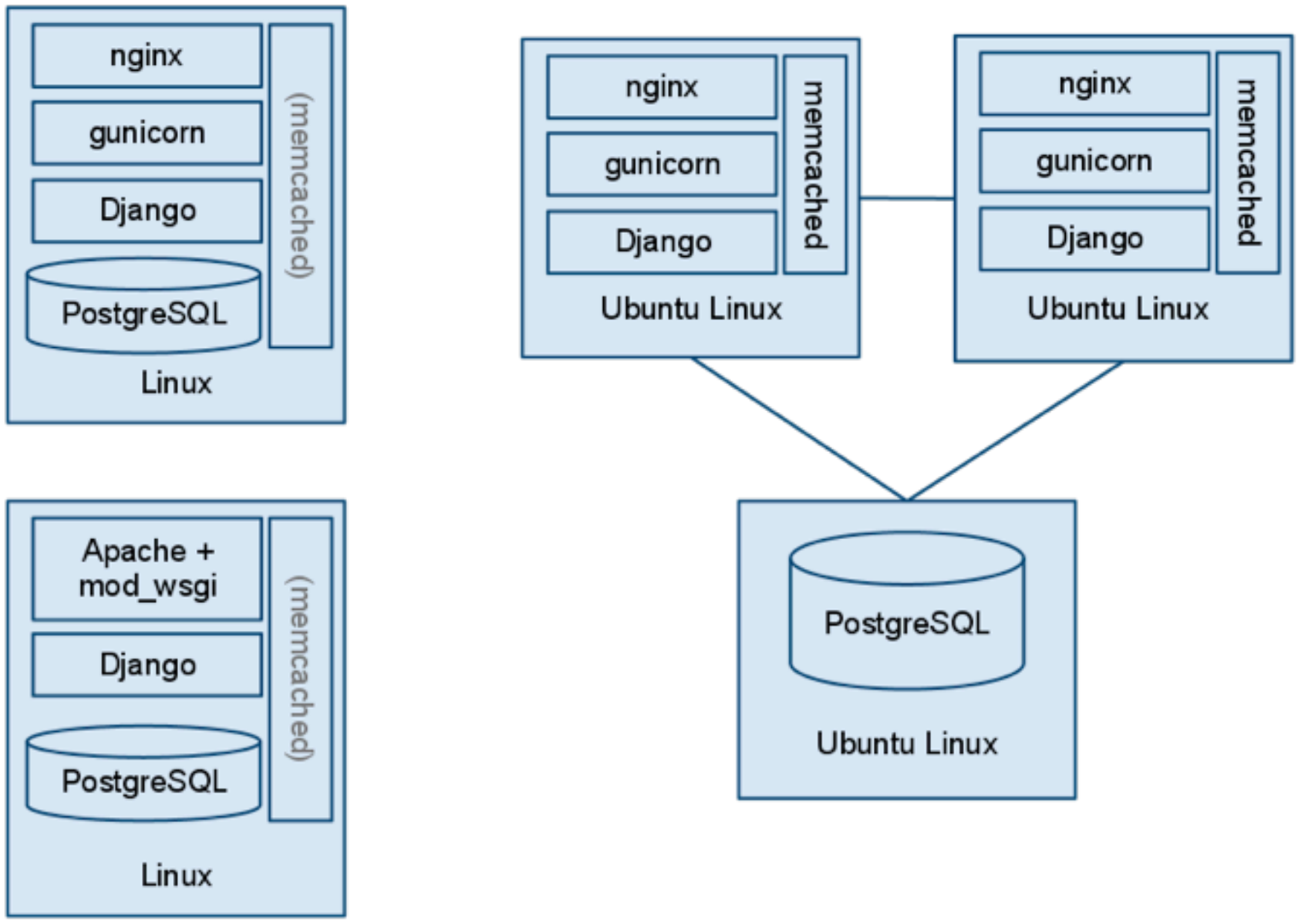
What is Django

- The Web Framework for Perfectionists with deadlines.
- MVC-inspired
- Features: ORM, Auto-admin, Clean URLs, Templating
- Community: Lots of awesome 3rd-party apps

Architecture



Stack


nginx
gunicorn
Django
PostgreSQL
(memcached)
Linux
nginx
gunicorn
Django
memcached
Ubuntu Linux
nginx
gunicorn
Django
memcached
Ubuntu Linux
PostgreSQL
Ubuntu Linux
Apache +
mod_wsgi
Django
PostgreSQL
(memcached)
Linux

Walkthrough

- Let's *quickly* build a simple twitter clone...
- *minitweet*

Prerequisites

- Python & Django is installed
  - Install Python: http://youtu.be/L5t5U0XnSew
  - Install Django: http://youtu.be/rIVwVOpwpsA
- You're OK working in a Terminal

Start a Project

- Defines an *instance* of django
- Configure Django options:
  - DB settings,
  - template locations,
  - applications

Start a Project

```
$ django-admin.py startproject minitweet
```

Start a Project

```
minitweet/
    __init__.py
    manage.py
    settings.py
    urls.py
```

DB Settings

```
DATABASES = {
    'default': {
        'ENGINE':'django.db.backends.sqlite3',
        'NAME': 'database.db',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    }
}
```

Project Templates

```
minitweet/
    __init__.py
    manage.py
    settings.py
    templates/
    urls.py
```

DB Settings

```
from os import path
_project_path =
path.dirname(path.abspath(__file__))

# ...
```

```
TEMPLATE_DIRS = (
    path.join(_project_path, "templates"),
)
```

Start an App

```
$ python manage.py startapp tweets
```

Start a Project

```
minitweet/
  tweets/
    __init__.py
    models.py
    tests.py
    views.py
```

Install your App

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    # 'django.contrib.admin',
    # Uncomment the next line to enable admin...
    # 'django.contrib.admindocs',
    'tweets',
)
```

Build your Models

```
from django.db import models

class Tweet(models.Model):
    content = models.CharField(max_length=140)
    posted_on = models.DateTimeField(auto_now_add=True)
```

Build your Models

```
from django.db import models
from django.contrib.auth.models import User

class Tweet(models.Model):
    content = models.CharField(max_length=140)
    posted_on = models.DateTimeField(auto_now_add=True)
    posted_by = models.ForeignKey(User)
```

Model Representation

```
class Tweet(models.Model):
    content = models.CharField(max_length=140)
    posted_on = models.DateTimeField(auto_now_add=True)
    posted_by = models.ForeignKey(User)

    def __unicode__(self):
        return self.content
```

Model Meta Data

```
class Tweet(models.Model):
    content = models.CharField(max_length=140)
    posted_on = models.DateTimeField(auto_now_add=True)
    posted_by = models.ForeignKey(User)

    def __unicode__(self):
        return self.content

    class Meta:
        ordering = ['-posted_on',]
        verbose_name = 'Tweet'
        verbose_name_plural = 'Tweets'
```

syncdb command

```
$ python manage.py syncdb
```

- Run when you create new models/install new apps.
- not really a *synchronize*.
- won't modify existing models
- FIRST TIME: set up an admin user
- If you want "migrations", use South: http://south.aeracode.org/

URLs

- Root URLconf:
  - `minitweet/urls.py`
- App URLConf:
  - `minitweet/tweets/urls.py`
- Map URLs to Views (functions)
- Use Regex to parse URL structure

Root URLconf

```
from django.conf.urls.defaults import patterns, include, url

# Uncomment the next two lines to enable the admin:
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('',
    url(r'^tweets/', include('tweet.urls')),

    # Uncomment the admin/doc line below to enable admin.
    #url(r'^admin/doc/',include('django.contrib.admindocs.urls')),
    # Uncomment the next line to enable the admin:
    # url(r'^admin/', include(admin.site.urls)),
)
```

Enable the Admin!

- Simple CRUD interface for data models
- Install django's admin contrib app
- Enable the admin URLs

Root URLconf

```
from django.conf.urls.defaults import patterns, include, url
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^tweets/', include('tweets.urls')),
    url(r'^admin/', include(admin.site.urls)),
)
```

Install the admin app

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'tweets',
)
```

Create your database

```
$ python manage.py syncdb
```

Try it all!

```
$ python manage.py runserver
  Validating models...

  0 errors found
  Django version 1.3.1, using settings 'minitweet.settings'
  Development server is running at http://127.0.0.1:8000/
  Quit the server with CONTROL-C.
```

Enable tweets admin

```
# minitweet/tweets/admin.py
from django.contrib import admin
import models

class TweetAdmin(admin.ModelAdmin):
    list_display = ('posted_by', 'posted_on')

admin.site.register(models.Tweet, TweetAdmin)
```

App URLconf

```
# minitweet/tweets/urls.py
from django.conf.urls.defaults import
patterns, include, url

urlpatterns = patterns('tweets.views',

    url(r'^$',
        'recent_tweets',
        name='tweets-recent_tweets'),
)
```

Views

- Just Python functions
- Contain arbitrary logic
- Accept an HTTP Request
- Return an HTTP Response
- Typically render content: HTML, JSON, PDF, CSV, Images, etc

```python
# minitweet/tweets/views.py
from django.contrib.auth.models import User
from django.shortcuts import render_to_response
from django.template import RequestContext

from models import Tweet

def recent_tweets(request):
    recent_tweets = Tweet.objects.all()

    template_data = {
        'recent_tweets': recent_tweets[:10],
    }
    template = "tweets/recent_tweets.html"
    ctx = RequestContext(request)

    return render_to_response(
        template,
        template_data,
        context_instance=ctx
    )
```

Templates

- Use the Django Template language: http://goo.gl/JLvQp
- Balance of Power & Ease (NOT python)
- CAN access python objects, attributes, methods
- **Filters**: Modify Presentation
- **Tags**: Encapsulate arbitrary logic
- Inheritance

Base Template

```
# minitweet/templates/base.html

<!DOCTYPE html>
<html>
<head>
    <title>
    {% block title %}{% endblock %}
    </title>
</head>
<body>
    {% block content %}{% endblock %}
</body>
</html>
```

App Templates

```
# minitweet/tweets/templates/tweets/recent_tweets.html

{% extends "base.html" %}

{% block title %}
Recent Tweets
{% endblock %}

{% block content %}
    <h1>Recent Tweets</h1>
    <ul>
    {% for t in recent_tweets %}
        <li>
        {{ t.posted_by }}<br/>
        {{ t.content }}
        </li>
    {% endfor %}
    </ul>
{% endblock %}
```

User Tweets

- Feature #2: Tweets from a single user

URLconf

```
# minitweet/tweets/urls.py

urlpatterns = patterns('tweets.views',

    url(r'^$',
        'recent_tweets',
        name='tweets-recent_tweets'),

    url(r'^(?P<username>.*)/$',
        'user_tweets',
        name='tweets-user_tweets'),
)
```

URLconf

```
# minitweet/tweets/urls.py

urlpatterns = patterns('tweets.views',

    url(r'^$',
        'recent_tweets',
        name='tweets-recent_tweets'),

    url(r'^(?P<username>.*)/$',
        'user_tweets',
        name='tweets-user_tweets'),
)
```

URLconf

```
# minitweet/tweets/urls.py

urlpatterns = patterns('tweets.views',

    url(r'^$',
        'recent_tweets',
        name='tweets-recent_tweets'),

    url(r'^(?P<username>.*)/$',
        'user_tweets',
        name='tweets-user_tweets'),
)
```

URLconf

```
# minitweet/tweets/urls.py

urlpatterns = patterns('tweets.views',

    url(r'^$',
        'recent_tweets',
        name='tweets-recent_tweets'),

    url(r'^(?P<username>.*)/$',
        'user_tweets',
        name='tweets-user_tweets'),
)
```

Views

```
# minitweet/tweets/views.py

def user_tweets(request, username):
    tweets = Tweet.objects.filter(
        posted_by__username=username)

    template_data = {
        'tweets': tweets[:10],
        'username': username,
    }
    template = "tweets/user_tweets.html"
    ctx = RequestContext(request)

    return render_to_response(
        template,
        template_data,
        context_instance=ctx
    )
```

Templates

 # minitweet/tweets/templates/tweets/user_tweets.html

```
{% extends "base.html" %}

{% block title %}
 {{ username }}'s Tweets
{% endblock %}
```

```
{% block content %}
    <h1>{{ username }}'s Feed</h1>
    <ul>
    {% for t in tweets %}
        <li>
        {{ t.content }}<br/>
        <em>Posted {{ t.posted_on|timesince }} ago</em>
        </li>
    {% endfor %}
    </ul>
{% endblock %}
```

Try it out!

- http://localhost:8000/tweets/brad/

Moving on...

- Feature #3: Posting tweets

URLconf

```
# add to: minitweet/tweets/urls.py

    url(r'^add/$',
        'add_tweet',
        name='tweets-add_tweet'),
```

Form Classes

- Know how to generate HTML
- Forms: Validate Content
- ModelForms: Validate Content for a Model
  - also create instances of a Model

Forms

```
# minitweet/tweets/forms.py

from django import forms
from models import Tweet

class TweetForm(forms.ModelForm):
    class Meta:
        model = Tweet
        fields = ('content', )
```

Views

```
# minitweet/tweets/views.py

# ...
from django.core.urlresolvers import reverse
# ...
from django.shortcuts import render_to_response, redirect
from forms import TweetForm

# ...
```

```python
# minitweet/tweets/views.py
def add_tweet(request):

    if request.method == "POST":
        # we'll come back to this...
    else:
        form = TweetForm()

    template_data = {
        "form": form,
    }
    template = "tweets/add_tweet.html"
    ctx = RequestContext(request)

    return render_to_response(
        template,
        template_data,
        context_instance=ctx
    )
```

```python
# minitweet/tweets/views.py
def add_tweet(request):

    if request.method == "POST":
        form = TweetForm(request.POST)
        if form.is_valid():
            tweet = form.save(commit=False)
            tweet.posted_by = request.user
            tweet.save()
            url = reverse("tweets-user_tweets",
                          args=[request.user])
            return redirect(url)
    else:

        # ...
```

Template

```
# minitweet/tweets/templates/tweets/add_tweet.html

{% extends "base.html" %}

{% block title %} Add Tweet {% endblock %}

{% block content %}
    <h1>Add Tweet</h1>

    <form action="{% url tweets-add_tweet %}" method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <p><input type="submit" value="Post Tweet!"/></p>
    </form>

{% endblock %}
```

Authentication

Whoops! Make sure only logged-in users can tweet:

```
from django.contrib.auth.decorators import login_required

# ... meanwhile, back in views.py ...

@login_required
def add_tweet(request):
    # ...
```

Enable Authentication

- Use Django's built-in auth app
- Already installed
- All the instructions are here:
  - https://docs.djangoproject.com/en/1.3/topics/auth/

Update Settings

```
# Add to minitweet/settings.py

LOGIN_URL = '/login/'
LOGOUT_URL = '/logout/'
LOGIN_REDIRECT_URL = '/tweets/'
```

Add some URLs

```
# minitweet/urls.py
urlpatterns = patterns('',
    url(r'^login/$',
     'django.contrib.auth.views.login', name='login'),
    url(r'^logout/$',
     'django.contrib.auth.views.logout', name='logout'),
    url(r'^tweets/', include('tweets.urls')),
    url(r'^admin/', include(admin.site.urls)),
)
```

Add a login template

```
# minitweet/templates/registration/login.html
{% extends "base.html" %}

{% block title %}Log In{% endblock %}

{% block content %}
    {% if user.is_authenticated %}
        <p>Welcome back, <strong>{{ user }}</strong></p>
    {% else %}
        <form action="{% url login %}" method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <p><input type="submit" value="Log In"/></p>
        </form>
    {% endif %}

{% endblock %}
```

Try it out!

http://localhost:8000/login/

Logging out

```
# minitweet/templates/registration/logged_out.html

{% extends "base.html" %}

{% block title %}Logged Out{% endblock %}

{% block content %}
    <p>Good Bye!</p>
{% endblock %}
```

Try it out!

http://localhost:8000/logout/

Where Next?

- https://docs.djangoproject.com/
- http://djangopackages.com/
- http://djangobook.com/en/2.0/ (a little dated)
- #django-users mailing list:
  - https://groups.google.com/forum/#!forum/django-users
- #django on irc

Thanks!