COMP47250: Team Software Project

Django Tutorial

Dr. Georgiana Ifrim
Insight Centre for Data Analytics
University College Dublin

04/06/2019

Outline

- Create Web App from scratch using Django
- Adding Some Functionality (e.g., news articles RSS scrapper)
- User Interface (HTML, CSS)
- Off-line vs Real-time Data Collection/Processing

Creating a Web App using Django

- What is Python?
  - An **interpreted, object-oriented, high-level programming language** [Source: https://www.python.org/doc/essays/blurb/]
  - ***"Programs must be written for people to read, and only incidentally for machines to execute."*** (**Harold Abelson**, **Structure and Interpretation of Computer Programs**)

- What is Django?
  - A free, open-source, **full stack Python Web framework.** Invented in 2005 by Web developers at the newspaper *Lawrence Journal-World.*

    [Source: https://www.djangoproject.com]
  - Examples sites that use Django: Pinterest (initially, now Flask), Instagram, The Washington Times
  - Django takes care of user authentication, content administration, site maps, etc. **Key idea**: a website is made of several components called applications.

- Example Apps created with Django

  [Source: http://elweb.co/33-projects-that-make-developing-django-apps-awesome/]

Install Python3.7: Anaconda

- **Anaconda:** Free Python distribution, includes popular Python packages for science, engineering, data analysis https://www.continuum.io/downloads

- Install Anaconda for **Python3.7** for your Operating System (or basic Miniconda): http://conda.pydata.org/docs/install/quick.html

- Once installed, go to the shell, check version of Python installed (Python3.7):

  python --version

- **Conda**: cross-platform package and environment manager. Create new **Python virtual environment** comp47250py37 and install packages. Run in shell:

conda create --name comp47250py37 python=3.7 numpy matplotlib scipy pandas scikit-learn django

Install Python3.7: Anaconda

- **Activate the newly created virtual environment:**

source activate comp47250py37

- **Install other required packages:**

conda install nltk

python -m nltk.downloader all (nltk needs some external corpora to work)

- **If package not available with conda, install with pip:**

pip install twitter

- **To deactivate this virtual environment:**

source deactivate

Pro TIP: Exporting a Virtual Environment

- If you work with Python and need to run your code/project later on, or on a new machine, or deliver it to someone, export your virtual environment using:

  pip freeze --all > pip-freeze-venv_name-project_name-date.txt

- **Example:**

  pip freeze --all > pip-freeze-venv_ comp47250py37-projectNL-040619.txt

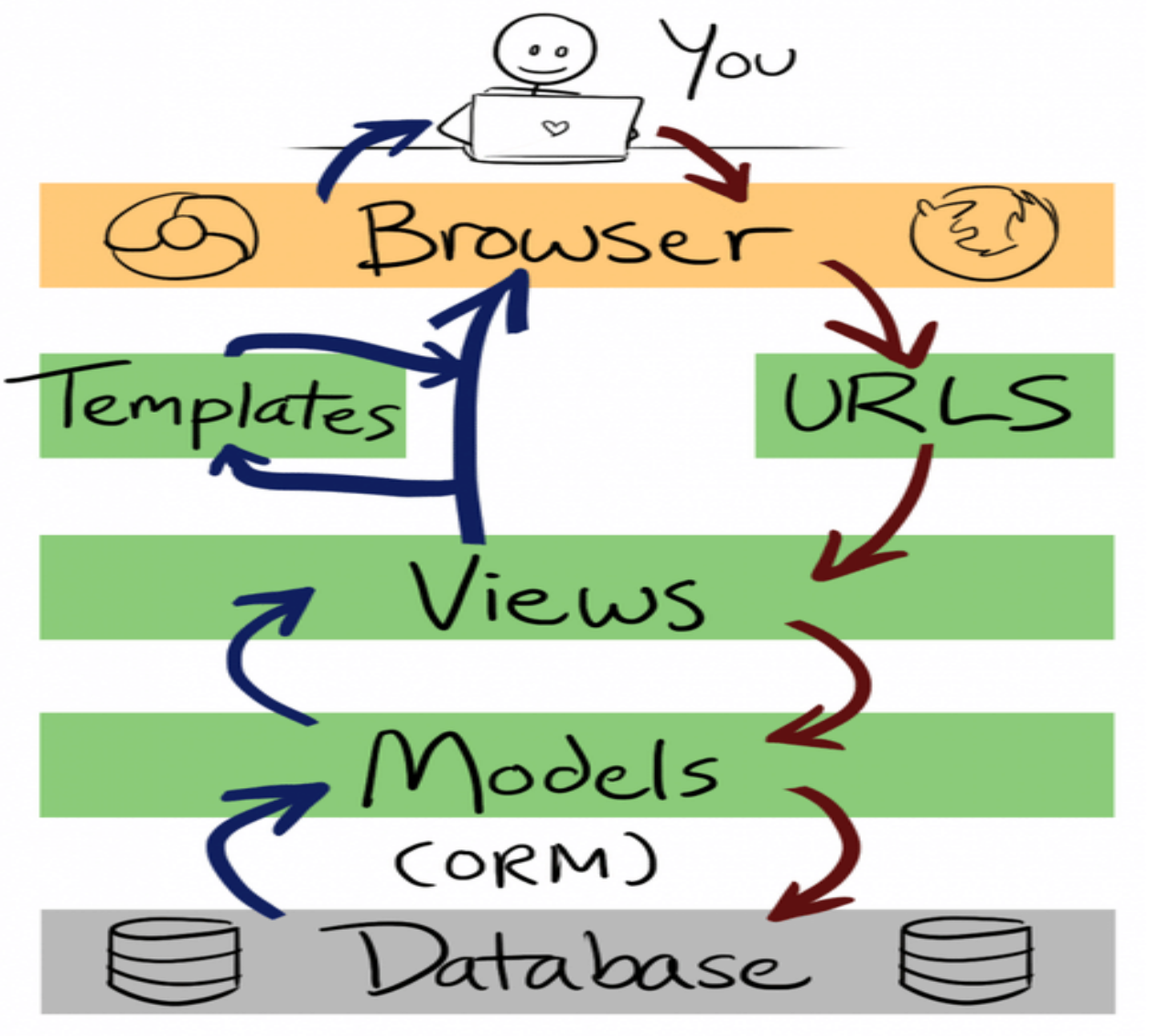
- This exports to a file all Python packages and their exact version installed at that time (when your code was still working!) in your virtual environment
- Look at the file created:

  less pip-freeze-venv_comp47250py37-projectNL-040619.txt
- **To setup the virtual environment on a new machine do:**

  pip install -r pip-freeze-venv_comp47250py37-projectNL-040619.txt

- You can call your exports file whatever you want, e.g., typical names are **requirements.txt**, required-packages.txt, etc.

Django High-Level Overview



The Big Picture – How Django is Structured

Django is modelled around a ***Model-View-Controller*** **(MVC)** framework. MVC is a software design pattern that aims to separate a web application into three interconnecting parts:

- The **model**, which provides the interface with the database containing the application data;
- The **view**, which decides what information to present to the user and collects information from the user;
- The **controller**, which manages the business logic for the application and acts as an information broker between the model and the view.

[Source: https://djangobook.com/tutorials/django-overview/]

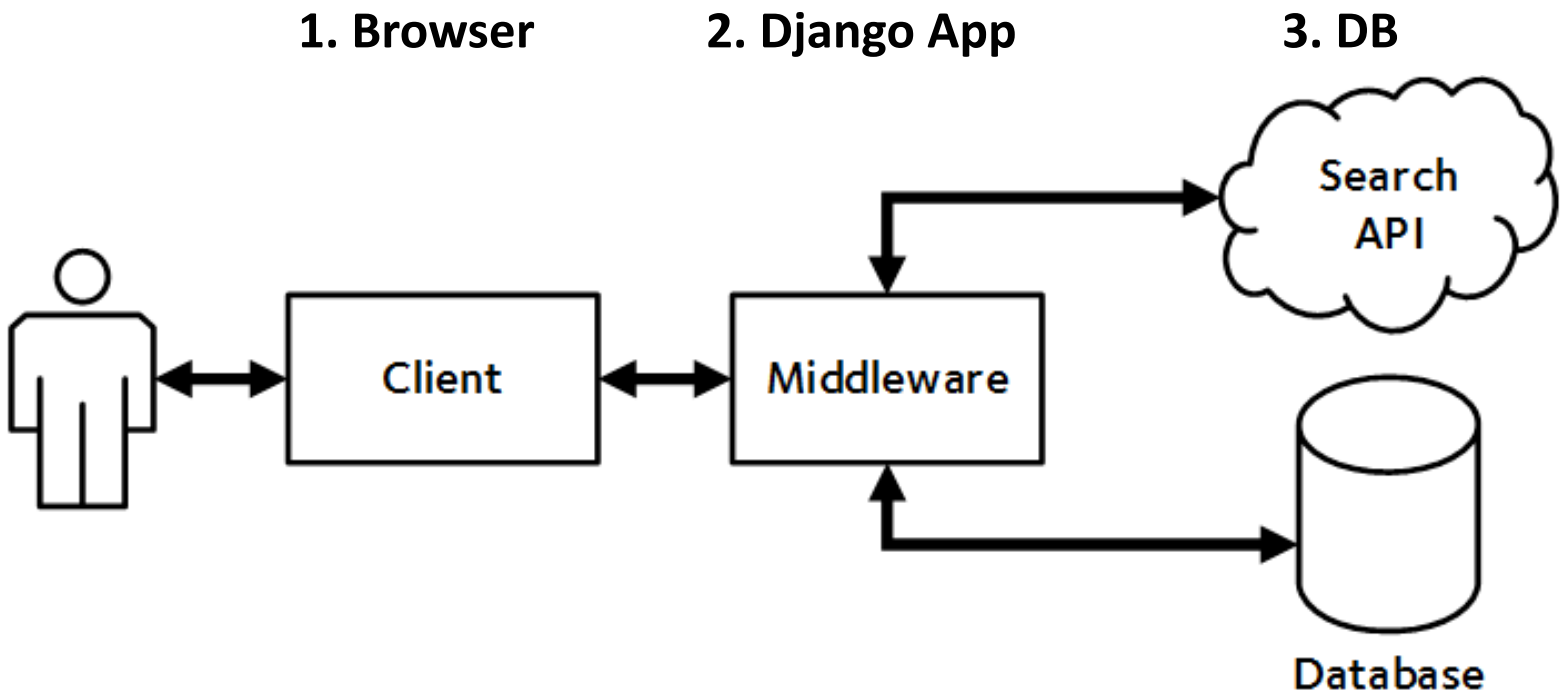
The Big Picture – How Django is Structured

Django uses slightly different terminology in its implementation of MVC:

- The **model** is functionally the same. Django's Object-Relational Mapping (ORM) provides the **interface to the application database**;
- The **template** provides display logic and is the **interface between the user and the Django application**;
- The **view** manages the bulk of the **applications data processing**, application logic and messaging.

[Source: https://djangobook.com/tutorials/django-overview/]

The Big Picture – How Django is Structured 3-Tier Architecture



[Source: https://djangobook.com/tutorials/django-overview/]

Setup: Install Django + libs

Assumes Python3 installed.

Install all required libs using your virtual environment (use **conda** or **pyvenv**)

Create a Python virtual environment for your app (e.g., **comp47250py37**):

conda create --name comp47250py37 python=3.7

Go to your venv to install all required libs:

source activate comp47250py37

Setup Libs and Database

Install libs using conda or pip (numpy, nltk, django, etc.):

conda install numpy sympy matplotlib scipy pandas scikit-learn

conda install django celery django-celery django-extensions psycopg2

pip install twitter feedparser beautifulsoup4

deactivate

(deactivate goes out of virtual environment to install PostgreSQL)

Install Postgresql DBMS, create user postgres and database newsdb (see last slides for Ubuntu vs Mac OS)**:**

createuser postgres (create an user **postgres**; params are platform dependent!)

createdb -O postgres newsdb (create a database called **newsdb**)

In terminal, look at newly created db using psql (can run SQL queries to look at tables):

psql -h localhost -U postgres newsdb

\dt (check what tables are in the database already)

\q (quit psql)

Building a Django App

source activate comp47250py37

Check Django is installed:

python -c "import django; print(django.get_version())''

Create a Django project:

django-admin startproject **newssite**

Look at created project structure (in terminal or IDE, e.g., PyCharm):

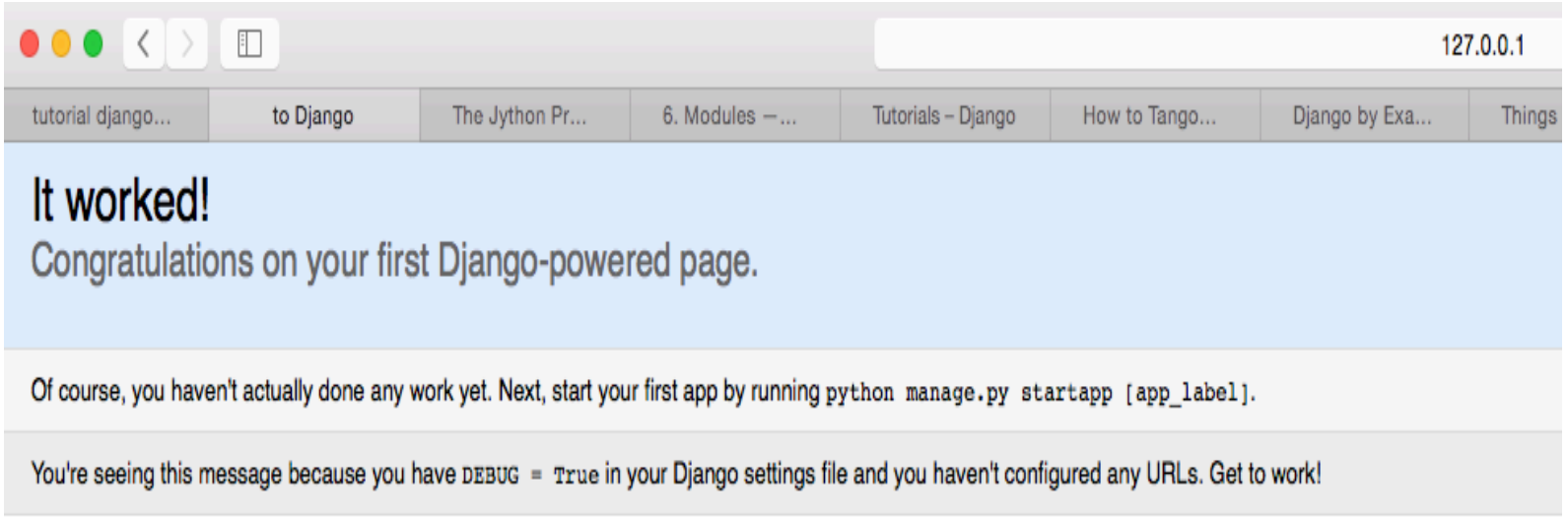
newssite/	Root directory, **container for your project**. Can rename it to whatever you want, e.g., **context4news** as in Github repo in refs
manage.py	**Command-line utility to interact with this Django project**
newssite/	**Python package for your project**
__init__.py	Tells Python this directory is a Python package.
settings.py	Settings/configuration for this Django project.
urls.py	A table of URLs of your Django-powered website.
wsgi.py	Required to get Django visible on the Web (e.g., by deploying with Apache and mod_wsgi).

Building a Django App

Run Django server from inside project folder **newssite/** (Terminal or PyCharm):

python manage.py runserver

Starting development server at http://127.0.0.1:8000/



An App for Articles

Create an app for articles (storage/collection/processing)

python manage.py startapp **articles**

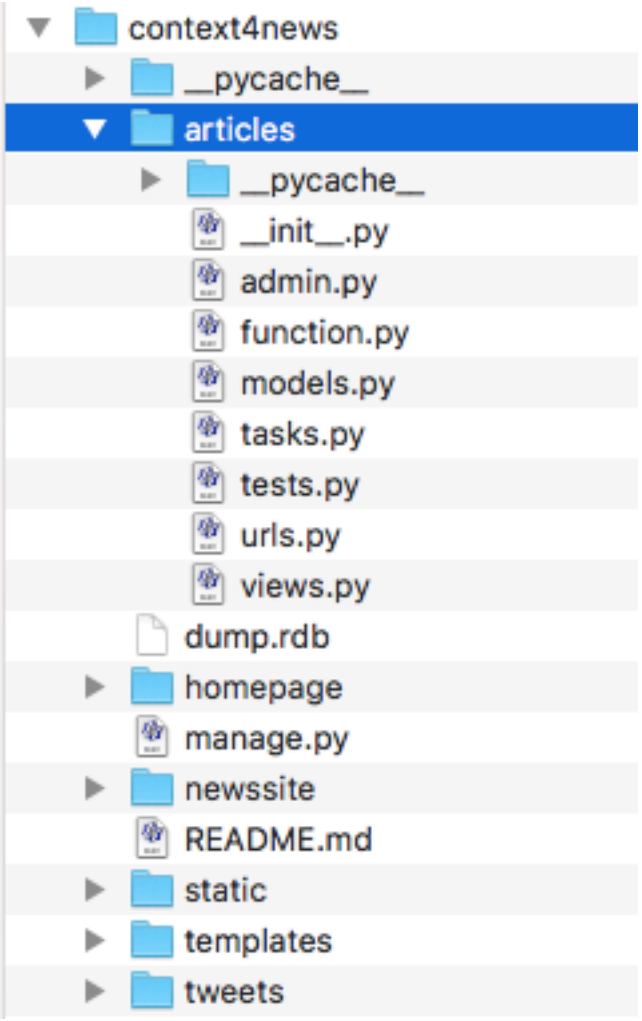
Need to manually add every app to the main Django project package, e.g.,: **newssite/settings.py**

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'articles')
```

```
# Application definition

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'djcelery',
    'django_extensions',
    'articles',
    'tweets',
    'homepage'
)
```

Django App Structure

Every Django App has a standard structure:



Django App Structure

Every Django App has a standard structure, e.g., if we look in folder **articles/**:

articles/

models.py	DB table(s) structure
views.py	functions to render content to webpages
urls.py	URLs of your app, each URL points to a view
function.py	data collection/processing (e.g., scrapRSS, parse HTML pages, store to DB)
tasks.py	cron-like re-occurring jobs (e.g., run scrapRSS every 10 mins)

HTML page for articles App in **newssite/templates:**

article_list.html	template html page, gets input from dedicated view (e.g., view named article_list())

Articles App: models

Storage (Database):

articles/models.py

Create db tables required by new App:

python manage.py makemigrations

python manage.py migrate

(Every time you install a new App need to repeat 'migration' steps above:

dropdb newsdb

createdb -O postgres newsdb

python manage.py migrate)

Look at your tables:

psql -h localhost -U postgres newsdb

\dt

```
import datetime

from django.db import models
from django.utils import timezone

# Create your models here.
class Article(models.Model):
    Headline = models.CharField(max_length=500, blank=True)
    SubHeadline = models.CharField(max_length=1000, blank=True)
    Url = models.URLField(max_length=400, unique=True, blank=True, null=True)
    DateTime = models.DateTimeField('date published', db_index=True)
    Keywords = models.CharField(max_length=5000)
    Content = models.TextField(blank=True)
    Type = models.CharField(max_length=100, db_index=True)
    Source = models.CharField(max_length=100)

    def was_published_recently(self):
        return self.DateTime >= timezone.now() - datetime.timedelta(days=1)

    def splitContent(self):
        return self.Content.split('\n')

    def getKeywords(self):
        return self.Keywords.split(",")

    def getSourceHeadline(self):
        return self.Source + ": " + self.Headline
```

Articles App: views

Views (Public Interface): articles/views.py

What are Views?

1. A **view** is a "type" of Web page in your Django application (serves a specific function and **has a specific template**).

2. In Django, web pages and other content are delivered by views.

3. Django will choose a **view** by examining the **URL** that's requested.

[Source: https://docs.djangoproject.com/en/1.8/intro/tutorial01/]

Articles App: views

Views (Public Interface): articles/views.py

Write your first view in **articles/views.py**

```
def article_index(request):
    return HttpResponse("Hello, world. You're at the articles index.")
```

```python
from django.shortcuts import render

# Create your views here.
import urllib
import json
import copy

from django.http import HttpResponse
from articles.models import Article

from django.shortcuts import render, get_object_or_404
from django.utils.timezone import utc
from datetime import timedelta, datetime


def article_index(request):
    return HttpResponse("Hello, world. You're at the articles index view.")
```

Articles App: urls

URLs for webpages of App: articles/urls.py

Create a URL for this view in **articles/urls.py**

```
urlpatterns = [
    url(r'^$', views.article_index, name=’article_index’)]
```

Update **newssite/urls.py** with App dependent URLs

```
urlpatterns = [
    url(r'^$', 'homepage.views.index', name='index'),
    url(r'^admin/', include(admin.site.urls)),
    url(r'^articles/', include('articles.urls')),
]
```

Articles App: templates

Templates:

newssite/templates

(html pages for UI,

mix of JS,

CSS, images)

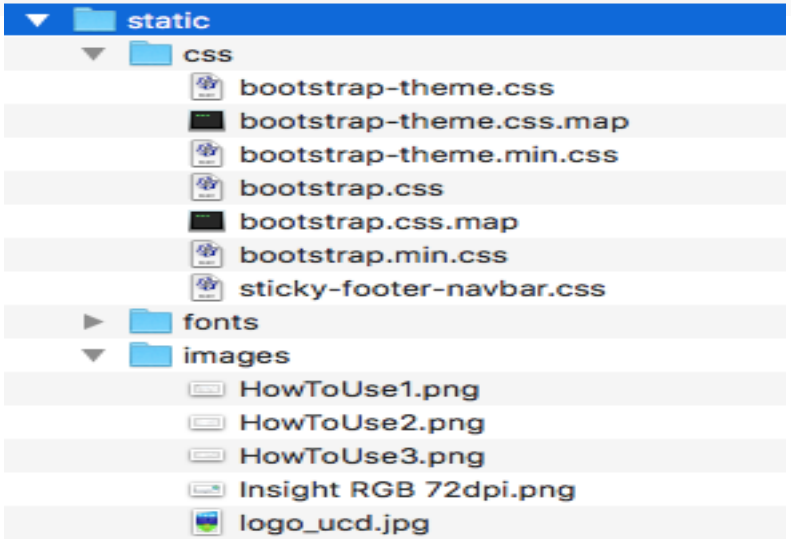
homepage.html

article_list.html

```
{% extends "base.html" %}
{% load staticfiles %}
{% block title %}
    DjangoNewsApp - Simple Django App for News
{% endblock %}



{% block body %}

    <article>
        <header class="page-header">
            <h2>Welcome to DjangoNewsApp!</h2>

        </header>
        <div class="col-md-10 col-md-offset-1" style="text-align: left">
            <h3>Description</h3>

            <p>
               DjangoNewsApp is an example app for news.</p>

        </div>
    </article>


{% endblock %}
```

newssite/static/

for images, CSS, JavaScript



Articles App: functions

Functionality: articles/function.py

E.g., add new rows to Article table, pre-process article content

```
import urllib
from datetime import datetime
from bs4 import BeautifulSoup
import nltk
import pytz

from articles.models import Article

def createArticleObject(title, subtitle, body, date, keywords, url, type, source):
    #print([title, subtitle, body, date, keywords, url, type, source])
    try:
        article = Article(Headline=title, SubHeadline=subtitle,
                          Content=body, Url=url,
                          DateTime=date, Keywords=keywords,
                          Type=type,
                          Source=source)
    except Exception as err:
                print("In createArticleObject():"+ err)
    return article

def createArticleByUrl(url):
    [title, subtitle, body, date, keywords, source] = getArticleDetailsByUrl(url)
    article = createArticleObject(title, subtitle, body, date, keywords, url, "RSS", source)
    return article



def getArticleDetailsByUrl(url):
    page = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(page)
    soup.prettify()
    title = soup.title.string
    #print("in getArticlesDetails(), title: "+ title + " Done!")


    title_clean = str.split(soup.title.string, ' | ')[0]
    doc_descr = soup.head.find("meta",attrs={"name":"description"}).get('content')
    doc_body = ''
```

Articles App: functions

Repeated jobs (cron like jobs):

articles/tasks.py

(Needs Redis + Celery)

pip install redis
redis-server &
python manage.py celeryd -l info -B -c 5

```
from datetime import timedelta, datetime
import sys
import urllib

from celery.schedules import crontab
from celery.task import periodic_task, task
from django.shortcuts import get_object_or_404
from django.utils.timezone import utc
import feedparser
from celery.exceptions import SoftTimeLimitExceeded, TimeLimitExceeded, Work
import redis

from articles.function import createArticleByUrl
from articles.models import Article

#read RSS feed every 15mins
#@periodic_task(run_every=crontab(minute='59,14,29,44'), time_limit=14 * 60,
@periodic_task(run_every=timedelta(minutes=30), expires=60)
def scrapAll():
    lock_id = "scrapAll"
    have_lock = False
    my_lock = redis.Redis().lock(lock_id, timeout=30 * 60)
    try:
        have_lock = my_lock.acquire(blocking=False)
        if have_lock:
            print(lock_id + " lock acquired!")
```

Data collection is independent of UI, e.g., run celery job in Terminal, and modify interface using PyCharm in parallel.

Apps for Articles & Tweets

Sample code for simple DjangoApp in Github public repo context4news.

git clone https://github.com/ucd-nlmsc-teamproject/context4news.git

Should give you a good start for reading RSS feeds and using a Restful API (Twitter).

Book: Tango with Django 1.9 (on Moodle)

Book gives details on high level design and specifications:

- System diagram (3-tier architecture)
- Wireframe
- Database model

Installation Steps: Ubuntu

Ubuntu: (assumes python3 already installed)

sudo apt-get install -y libblas-dev liblapack-dev gfortran

sudo apt-get install -y libfreetype6-dev libpng-dev

sudo apt-get install -y libpq-dev

sudo apt-get install apache2

sudo apt-get install libapache2-mod-wsgi-py3

sudo apt-get install rabbitmq-server

sudo apt-get install build-essential

sudo apt-get install python3.4-dev

sudo apt-get install libxft-dev

Downloads latest setup tools, and installs pip (need to be in venv):

curl https://bootstrap.pypa.io/ez_setup.py -o - | python

cd setuptools-16.0

python ez_setup.py

easy_install pip

pip install numpy sympy matplotlib scipy pandas scikit-learn

Install libs using pip (nltk for python3, django, etc.):

pip install django celery django-celery django-extensions psycopg2

pip install twitter feedparser beautifulsoup4

deactivate

Install Postgresql (with user “postgres”, passwd “postgres”, database “newsdb”)

sudo apt-get install postgresql postgresql-contrib

createuser postgres

createdb newsdb

Look at tables in db using “psql”:

psql -h localhost -U postgres -d newsdb

\l

\dt

Installation Steps: Mac OS

Mac: (if not using Anaconda you can install python with brew; recommended to use Anaconda)
brew install python3
pyvenv-3.4 newsapp
source newsapp/bin/activate
pip install numpy sympy matplotlib scipy pandas scikit-learn
pip install -U nltk
pip install django celery django-celery django-extensions
pip install twitter feedparser beautifulsoup4 fastcluster
pip install psycopg2
pip install django-preserialize redis

Install Postgresql:
brew install postgresql
ln -sfv /usr/local/opt/postgresql/*.plist ~/Library/LaunchAgents
launchctl load ~/Library/LaunchAgents/homebrew.mxcl.postgresql.plist

Start Postgresql manually:
pg_ctl -D /usr/local/var/postgres -l /usr/local/var/postgres/server.log start
createuser -d -P postgres
createdb –U postgres newsdb
Look at tables in db using "psql":
psql -h localhost -U postgres -d newsdb
\dt

References

- https://www.python.org/doc/essays/blurb/
- https://docs.djangoproject.com/en/1.9/intro/tutorial01/
- https://docs.djangoproject.com/en/2.0/intro/tutorial01/
- http://elweb.co/33-projects-that-make-developing-django-apps-awesome/

- Online tutorials:
- https://www.youtube.com/watch?v=cvceyNioank
- https://www.quora.com/How-do-I-find-a-good-Django-tutorial-for-beginners
- http://tutorial.djangogirls.org/en/
- https://www.digitalocean.com/community/tutorials/how-to-run-django-with-mod_wsgi-and-apache-with-a-virtualenv-python-environment-on-a-debian-vps
- https://www.quora.com/Angular-js-is-the-hottest-JavaScript-client-side-framework-What-server-side-back-end-web-technology-goes-best-with-Angular
- http://conda.pydata.org/docs/using/envs.html#remove-an-environment
- https://wiki.python.org/moin/WebFrameworks
- https://realpython.com/blog/python/python-web-applications/