

# Django

Sears Merritt

CSCI 5448: Object-Oriented Analysis  
and Design

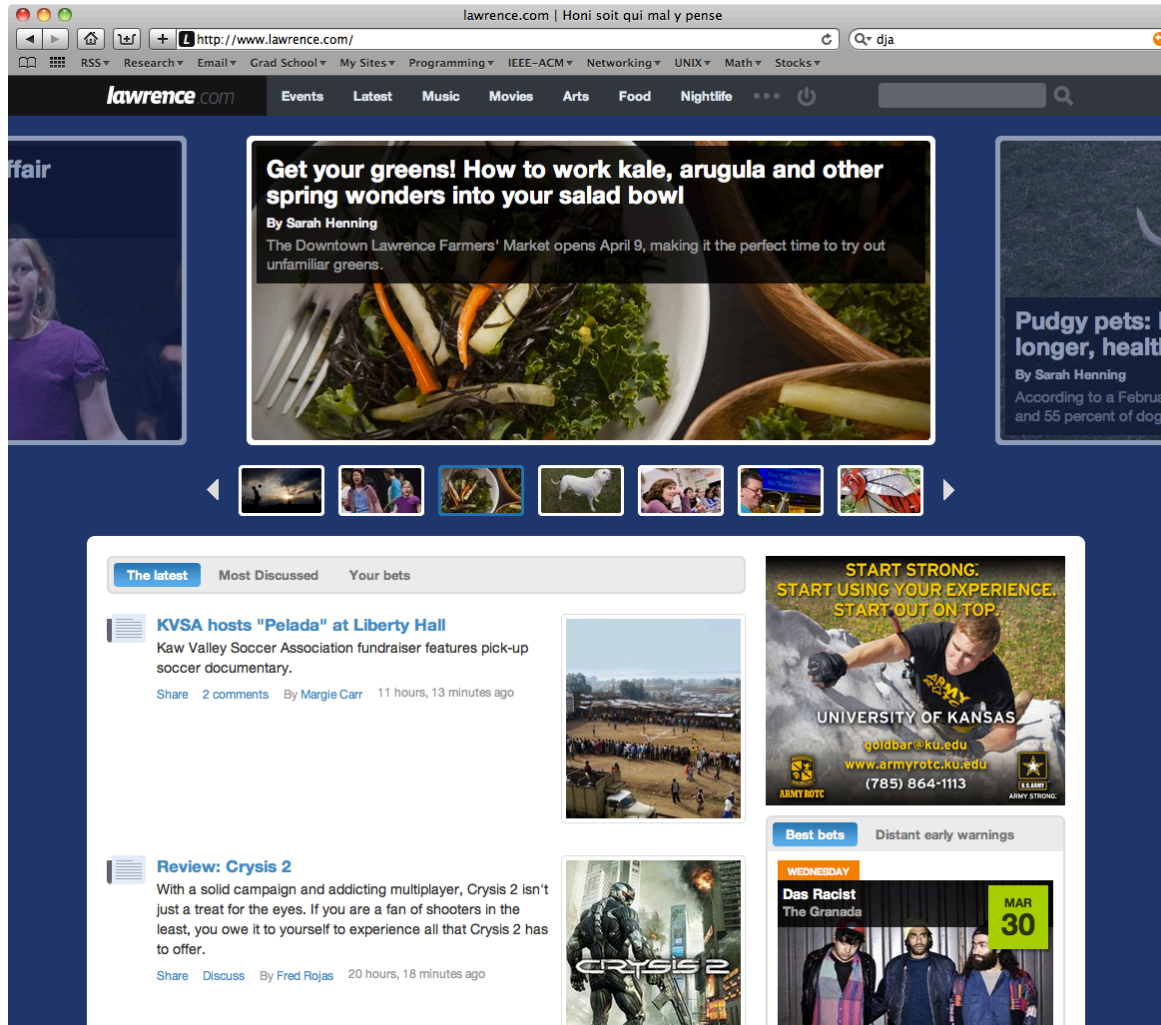
# What is Django?

“Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.” – [Django Website](#)

# History

- Began as a framework for The World Company, used to manage their news sites.
- Developers needed an efficient method for managing and publishing news articles.
- Open-sourced under BSD license in July 2005
- Django Software Foundation founded in June 2008 to continue the management and support of the framework

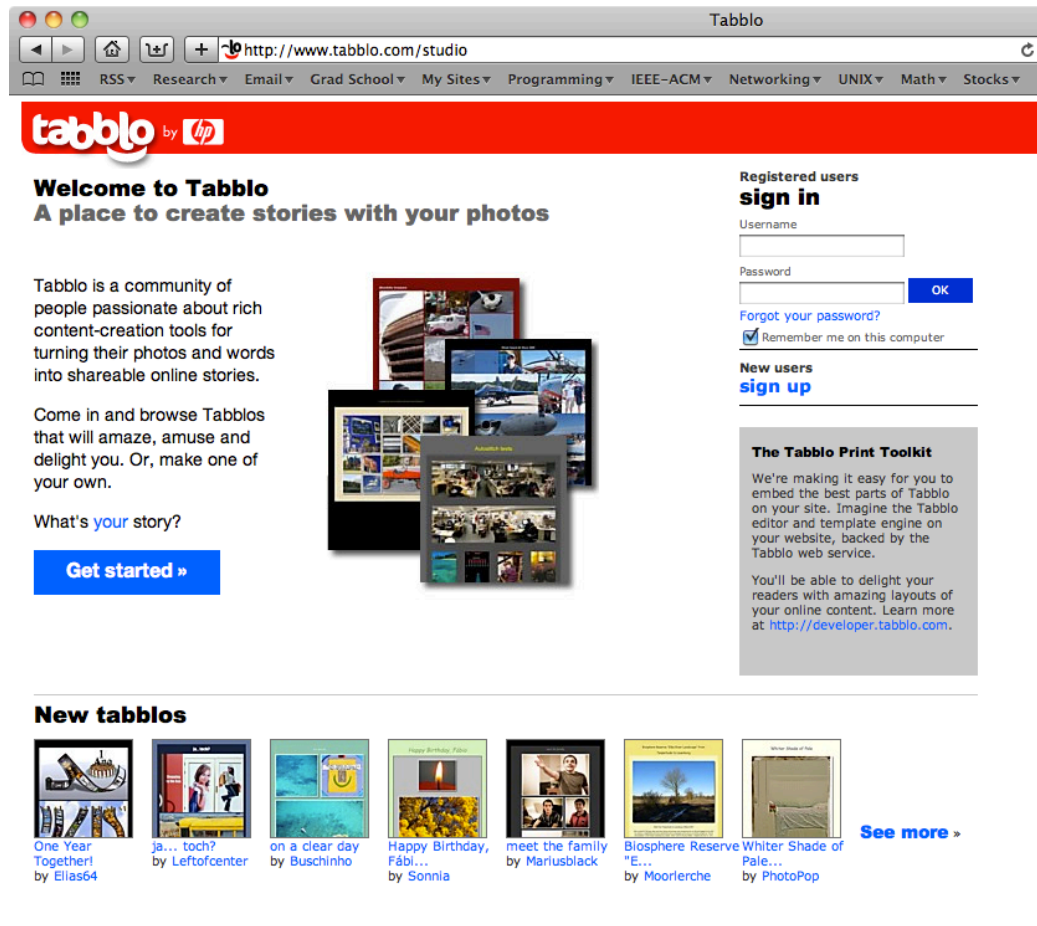
# Django Driven Sites: www.lawrence.com



# Django Driven Sites: www.washingtonpost.com

The screenshot shows a web browser displaying the 'U.S. Congress Votes Database' page from The Washington Post. The browser's address bar shows the URL 'http://projects.washingtonpost.com/congress/112/'. The page features a navigation menu with categories like 'POLITICS', 'OPINIONS', 'LOCAL', 'SPORTS', 'National', 'World', 'Business', 'Investigations', 'Lifestyle', 'Entertainment', and 'Multimedia'. Below the navigation, there's a search bar and a section titled 'In the News' with links to various news items. A prominent advertisement for Fisher Investments is displayed, titled 'Is there a crisis in muni bonds?'. The main content area is divided into two columns. The left column is titled 'The U.S. Congress Votes Database' and includes sections for 'Explore the 112th Senate', 'SENATE PARTY LEADERS' (listing Harry Reid and Mitch McConnell), 'Explore the 112th House', and 'HOUSE PARTY LEADERS' (listing John Boehner and Nancy Pelosi). The right column is titled 'Recent key votes' and lists several legislative actions, such as 'To stop federal funding of National Public Radio' (PASSED), 'Democratic Senate budget plan' (REJECTED), 'GOP House budget bill' (DEFEATED), and 'Two-week budget bill' (PASSED). A large advertisement on the right side of the page is titled '8 investing mistakes you should avoid in 2011' and promotes a guide by Ken Fisher.

# Django Driven Sites: www.tabblo.com



The screenshot shows a browser window with the URL <http://www.tabblo.com/studio>. The page features a red header with the "tabblo by hp" logo. The main content area includes a welcome message, a sign-in form for registered users, a sign-up link for new users, and a section for "New tabblors" with a grid of featured stories. A "See more" link is positioned to the right of the grid.

**Welcome to Tabblo**  
A place to create stories with your photos

Tabblo is a community of people passionate about rich content-creation tools for turning their photos and words into shareable online stories.

Come in and browse Tabblors that will amaze, amuse and delight you. Or, make one of your own.

What's [your](#) story?

[Get started »](#)

**Registered users**  
**sign in**

Username

Password

[Forgot your password?](#)

Remember me on this computer








**New users**  
[sign up](#)

**The Tabblo Print Toolkit**

We're making it easy for you to embed the best parts of Tabblo on your site. Imagine the Tabblo editor and template engine on your website, backed by the Tabblo web service.

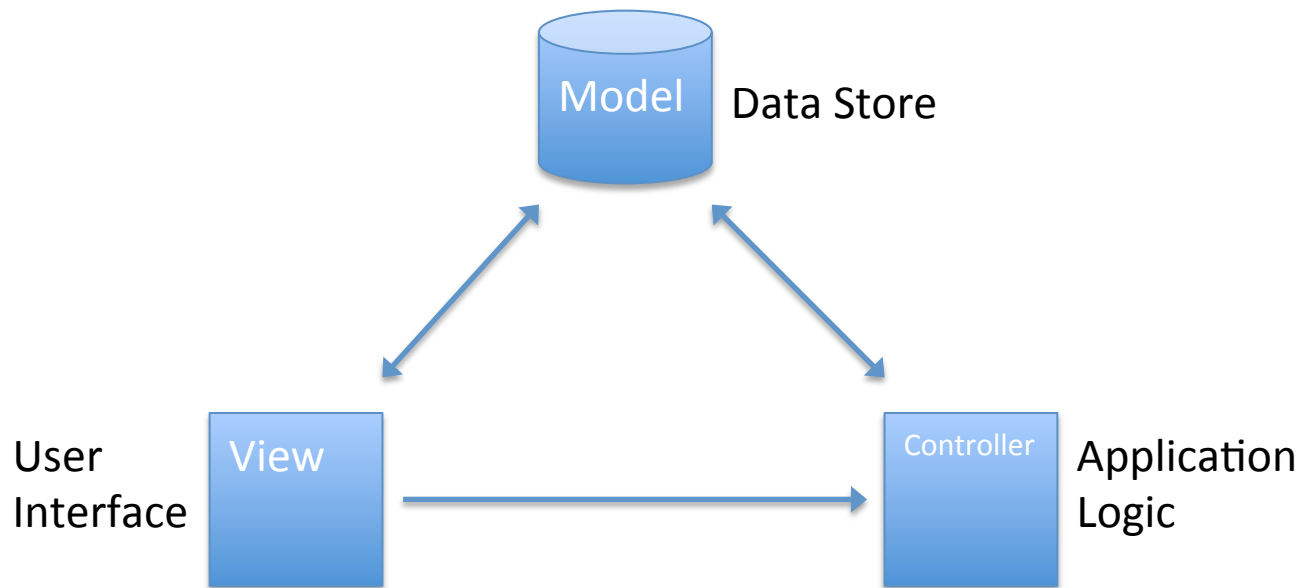
You'll be able to delight your readers with amazing layouts of your online content. Learn more at <http://developer.tabblo.com>.

**New tabblors**

 One Year Together! by Elias64	 ja... toch? by Leftofcenter	 on a clear day by Buschinho	 Happy Birthday, Fábl... by Sonnia	 meet the family by Mariusblack	 Biosphere Reserve Whiter Shade of Pale... by Moorlerche	 Whiter Shade of Pale... by PhotoPop
---	---	---	---	---	---	---

[See more »](#)

# Primary Concept and Design Pattern: Model-View-Controller



# Django Model-View-Controller Overview

- **Model:** Holds data, typically in a database. Superclass is Model. Models for applications are created in Django by sub-classing the Model class. A variety of databases are supported ranging from sqlite3 to mysql that are used to store tables of model objects.
- **View:** Presents data to users using Django templates and template language. Accessed by creating html pages with Django template language. Content is dynamically rendered using a wide range of Django methods.
- **Controller:** Controls which data and views are presented. Accessed via routing requests to particular view routines. Each view routine has access to the models and additional logic to create and manipulate data. Request routing is handled through urls.py while view logic is handled through functions defined in views.py



# Model

- Application models stored in models.py
- New models are subclassed from `django.db.models.Model`
- Models are composed of different types of fields:
  - Characters
  - Integers
  - Floats
  - Dates
  - And more!

```
from django.db import models

class Symptom(models.Model):
    symptom = models.CharField(max_length=200)
    def __unicode__(self):
        return self.symptom

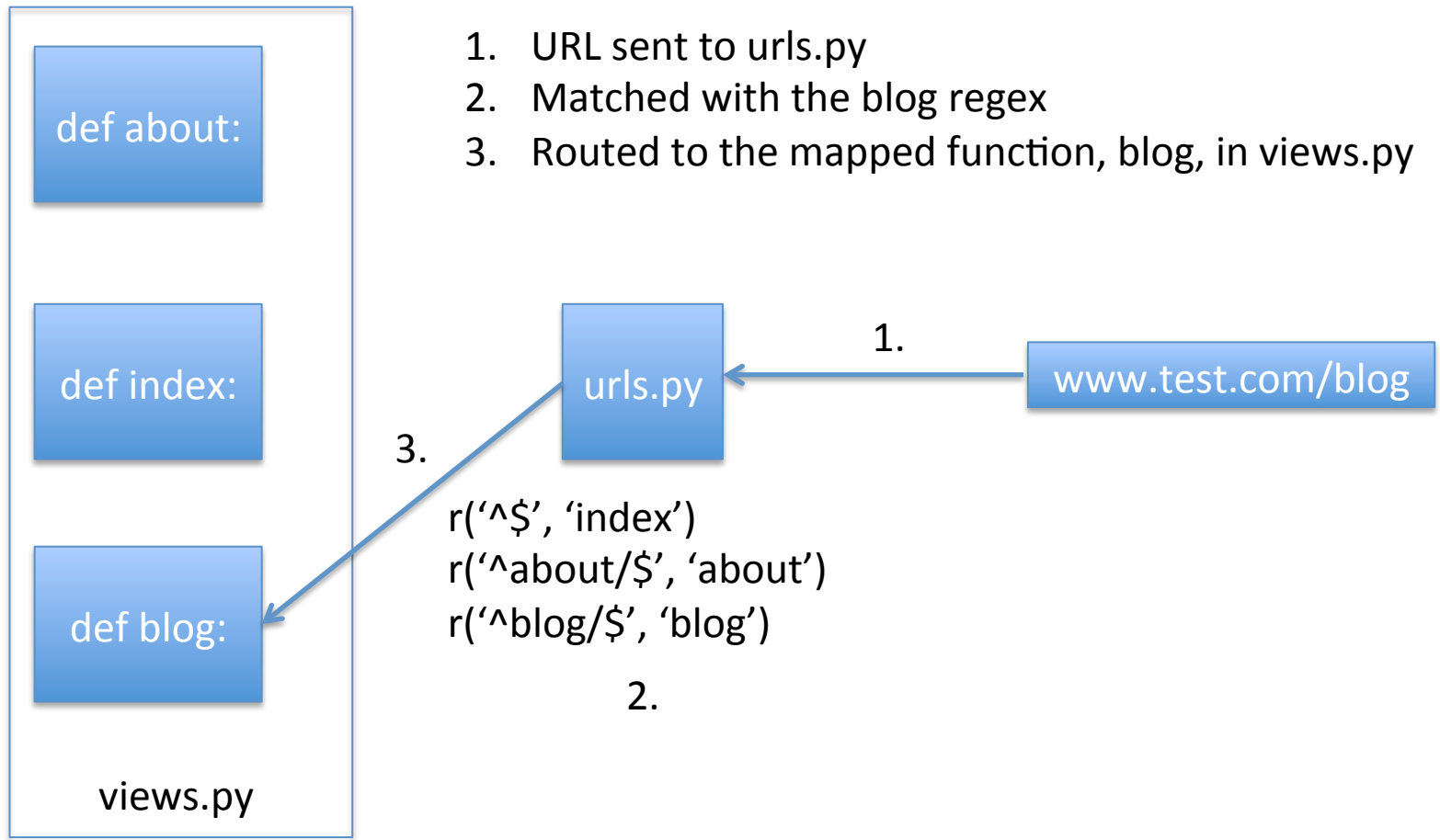
class Location(models.Model):
    city = models.CharField(max_length=200)
    state = models.CharField(max_length=200)
    latitude = models.FloatField()
    longitude = models.FloatField()
    pop = models.PositiveIntegerField(default=1)
    def __unicode__(self):
        return ', '.join((self.city, self.state))

class UserReport(models.Model):
    symptom = models.CharField(max_length=2400)
    latitude = models.FloatField()
    longitude = models.FloatField()
    timestamp = models.DateTimeField()
    location = models.ForeignKey(Location)
```

# View Controllers

- View controllers are invoked by routing requests to them via `urls.py`.
- `Views.py` defines the set of controllers associated with each rule in `urls.py`
- Each view can:
  - Accept HTTP messages
  - Handle input and output
  - Generate dynamic content for rendering by the template language

# View Controller Routing



# View Controllers – urls.py

- List of regular expressions matching urls with view routines
- Left hand side is regex, right side is function in views.py

```
from django.conf.urls.defaults import *  
  
from django.contrib import admin  
admin.autodiscover()  
  
urlpatterns = patterns('website.socialhealth.views',  
    (r'^$', 'index'),  
    (r'^results/$', 'results'),  
    (r'^report/$', 'report'),  
    (r'^result/$', 'result'),  
)
```

# View Controllers – views.py

- Each view is defined as a function
- Each function receives a request
- A request contains dictionaries of data
- Using requests, data can be accessed easily for saving or generating dynamic content
- Each view returns an HTTP response ranging from redirects to dynamically rendered HTML pages

# View Controllers – views.py

```
from django.template import Context, loader
from django.template import RequestContext
from django.shortcuts import render_to_response, get_list_or_404, redirect
from website.socialhealth.models import Symptom, UserReport, Location

def index(request):
    return render_to_response('index.html', context_instance=RequestContext(request))

def report(request):
    sympList = get_list_or_404(Symptom)
    return render_to_response('report.html', {'sympList': sympList}, \
        context_instance=RequestContext(request))

def results(request):
    return render_to_response('results.html', context_instance=RequestContext(request))

def result(request):
    if request.POST:
        r = UserReport()
        r.email = request.POST.get('email')
        r.symptom = r.symptom.lstrip(',')
        r.timestamp = datetime.datetime.fromtimestamp(time.time())
        r.longitude = request.POST.get('longitude')
        r.latitude = request.POST.get('latitude')
        r.location = Location.objects.get(city=cluster((float(r.latitude), float(r.longitude))))
        r.save()
        return redirect('/')
    else:
        return redirect('/')
```

# Views – Templates and Template Language

- Views are essentially html pages with embedded template language
- Template language is Django specific
- Django renders template language using data received from the view controller

# Template Language Syntax

Template language consists of intuitively tagged python statements

```
{% for item in list %}  
    {{ item }}  
{% endfor %}
```

```
{ % if item %}  
    {{ item }}  
{% endif %}
```

This language is embedded in html pages and gets rendered when a view controller passes data and the page to the renderer

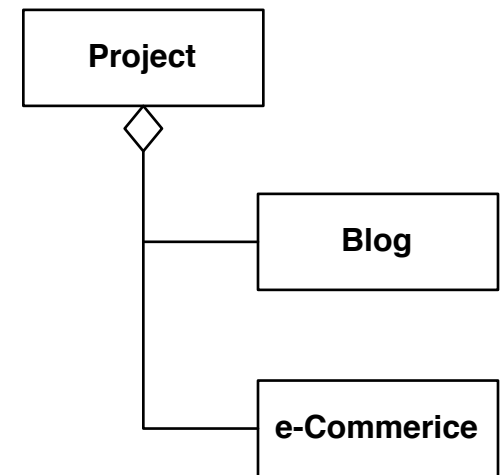


# Views – Template and Template Language

```
if sympList %}
form id="form" action="../result/" method="post">
csrf_token %}
for symptom in sympList %}
  <input type="checkbox" name="symptom{{ forloop.counter }}" id="symptom{{ forloop.counter }}" value="{{ symptom }}" />
  <label for="symptom{{ forloop.counter }}">{{ symptom }}</label><br />
endfor %}
  <input type="hidden" name="latitude" id="latitude" value="" />
  <input type="hidden" name="longitude" id="longitude" value="" />
  <input type="hidden" name="email" id="email" value="null" />
  <input type="submit" name="submit" id="submit" value="Submit" />
form>
else %}
<p>No symptoms are available.</p>
endif %}
```

# Django Projects and Applications

- Django organizes MVC implementations into projects.
- Projects aggregate applications.
- Each application has its own models, views, and controllers.



# Project Configuration – settings.py

- Holds the configuration settings for a project
  - Database configuration
  - Media directory (css, javascript, etc.)
  - Template directory (html files)
  - Django middleware (such as authentication and CSRF)
  - Django modules
  - Project applications

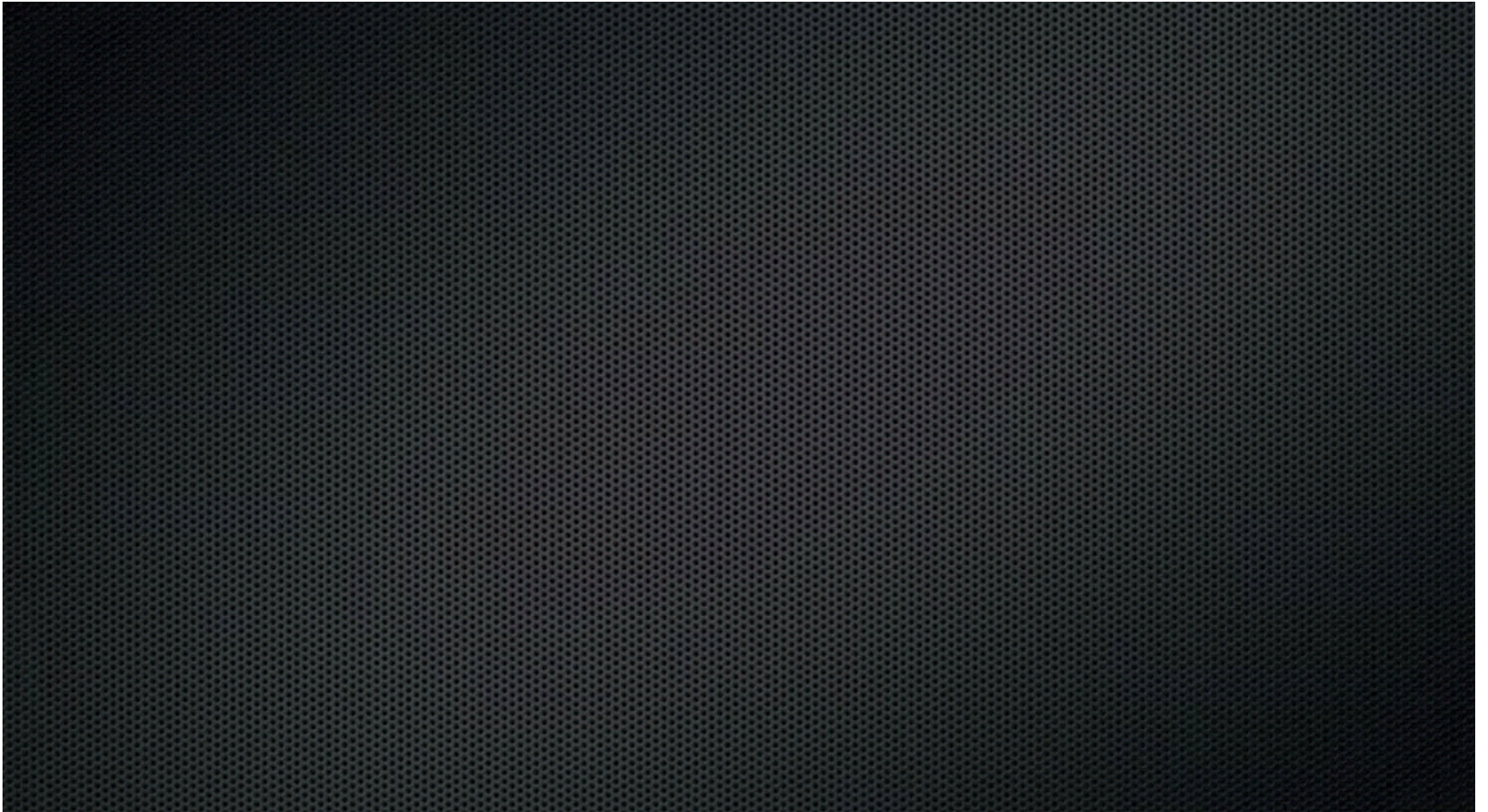
# Creating a Django Website

- Create a project
  - `django-admin.py startproject myproject`
- Create an app
  - `django-admin.py startapp myapp`
- Configure `settings.py`
- Configure the MVC implementation for myapp
  - `urls.py`
  - `views.py`
  - `models.py`
- There is an excellent tutorial [here](#).

# The Life of a Request

1. HTTP packet arrives at web server
2. WSGI routes packet to Django core
3. Django core:
  1. parses message
  2. routes the message to the appropriate view handler via `urls.py`
  3. Creates a request object
4. Controller handler parses request and performs appropriate logic via handlers in `views.py`
5. Controller hands resulting data to the template system
6. Template system pulls HTML, fills with content from view handler, sends response

# Django in Action



# References

- [Django Website](#)
- [Django Wikipedia](#)
- [Social Health Website](#)