

# Introduction to

```
from django.shortcuts import render_to_response from django.http import HttpResponseRedirect, HttpResponse from django.core.urlresolvers import reverse from django.template import RequestContext, loader
def get_object_or_404(obj, pk=pk): try: selected_choice = p.choice_set.get(pk=request.POST['choice']) except (KeyError, Choice.DoesNotExist): return render_to_response('polls/detail.html', { 'poll': p, 'error_message': "You didn't select a choice.", }, context_instance=RequestContext(request)) else: selected_choice.votes += 1 selected_choice.save() return HttpResponseRedirect(reverse('polls:results', args=(p.id,))) from django.utils import unittest from myapp.models import Animal class AnimalTestCase(unittest.TestCase): def setUp(self): self.objects.create(name="lion", sound="roar") self.cat = Animal.objects.create(name="cat", sound="meow") def test_animals_can_speak(self): self.assertEqual(self.lion.speak(), 'The lion says "roar"') self.assertEqual(self.cat.speak(), 'The cat says "meow"') from django.contrib.syndication.views import FeedDoesNotExist from django.shortcuts import get_object_or_404 class BeatFeed(Atom1Feed): feed_template = 'feeds/beat_description.html' def get_object(self, request, beat_id): return get_object_or_404(Beat, pk=beat_id) def title(self, obj): return "Chicagocrime.org: Crimes for beat %d" % obj.beat def link(self, obj): return obj.get_absolute_url() def description(self, obj): return "Crimes recently reported in police beat %d" % obj.beat def items(self, obj): return Crime.objects.filter(beat=obj).order_by('-time_date')[ :30] from django.utils.html import conditional_escape from django.utils.safestring import mark_safe @register.filter(needs_autoescape=True) def initial_letter_filter(text, autoescape=None): first, other = text[0], text[1:] if autoescape: esc = conditional_escape else: esc = lambda x: x result = '<strong>%s</strong>%s' % (escfirst, esclother) return mark_safe(result) from django import template def do_current_time(parse, token): try: tag_name, format_string = token.split_contents(), except ValueError: raise template.TemplateSyntaxError("%r tag requires a single argument" % token.contents.split()[0]) if not (format_string[0] == format_string[-1] and format_string[0] in ('"', '')): raise template.TemplateSyntaxError("%r tag's argument should be in quotes" % tag_name) return CurrentTimeNode(format_string[1:-1]) from django import template import datetime class CurrentTimeNode(template.Node): def __init__(self, format_string): self.format_string = format_string def render(self, context): return datetime.datetime.now().strftime(self.format_string) class CycleNode(Node): def __init__(self, cyclevars): self.cyclevars = cyclevars def render(self, context): if self not in context.render_context: context.render_context[self] = itertools.cycle(self.cyclevars) cycle_iter = context.render_context[self] return cycle_iter.next() class MaleManager(models.Manager): def get_query_set(self): return super(MaleManager, self).get_query_set().filter(sex='M') class FemaleManager(models.Manager): def get_query_set(self): return super(FemaleManager, self).get_query_set().filter(sex='F') class Person(models.Model): name = models.CharField(max_length=50) last_name = models.CharField(max_length=50) sex = models.CharField(max_length=1, choices=(('M', 'Male'), ('F', 'Female')) people = models.Manager() maleManager() women = FemaleManager() from django import forms class ContactForm(forms.Form): subject = forms.CharField(max_length=100) message = forms.CharField(max_length=100) sender = forms.EmailField(required=True) cc_myself = forms.BooleanField(required=False) if form.is_valid(): subject = form.cleaned_data['subject'] message = form.cleaned_data['message'] sender = form.cleaned_data['sender'] cc_myself = form.cleaned_data['cc_myself'] recipients = ['info@example.com'] if cc_myself: recipients.append(sender) from django.core.mail import send_mail send_mail(subject, message, sender, recipients) return HttpResponseRedirect('/thanks/') from django.http import HttpResponse from django.template import Context, loader def my_view(request): t = loader.get_template('myapp/template.html') c = Context({'name': 'John'}) return HttpResponse(t.render(c), mimetype="application/xhtml+xml") from django.template.loaders import app_directories class Loader(app_directories.Loader): is_usable = True def load_template_source(template_name, template_dirs=None): source, origin = self.load_template_source(template_name, template_dirs) template = Template(source) return template, origin from django import template register = template.Library() @register.filter @stringfilter def lower(value): return value.lower() from django.utils.html import conditional_escape from django.utils.safestring import mark_safe @register.filter(needs_autoescape=True) def initial_letter_filter(text, autoescape=None): first, other = text[0], text[1:] if autoescape: esc = conditional_escape else: esc = lambda x: x result = '<strong>%s</strong>%s' % (escfirst, esclother) return mark_safe(result) from django import template def do_current_time(parse, token): try: tag_name, format_string = token.split_contents(), except ValueError: raise template.TemplateSyntaxError("%r tag requires a single argument" % token.contents.split()[0]) if not (format_string[0] == format_string[-1] and format_string[0] in ('"', '')): raise template.TemplateSyntaxError("%r tag's argument should be in quotes" % tag_name) return CurrentTimeNode(format_string[1:-1]) from django.conf import settings from django.contrib.auth.models import User, check_password class SettingsBackend(object): supports_inactive_user = False def authenticate(self, username=None, password=None): login_val = getattr(settings, 'ADMIN_LOGIN', None) pwd_valid = check_password(password, settings.ADMIN_PASSWORD) if login_val and pwd_valid: try: user = User.objects.get(username=username) user.DoesNotExist: user = User(username=username, password=getattr(settings, 'PASSWORD', None)) user.is_staff = True user.is_superuser = True user.save() return user return None def get_user(self, user_id): try: user = User.objects.get(pk=user_id) except User.DoesNotExist: return None class EntryDetail(models.Model): entry = models.OneToOneField(Entry) details = models.TextField() class Blog(models.Model): name = models.CharField(max_length=100) tagline = models.TextField() def __unicode__(self): return self.name class Author(models.Model): name = models.CharField(max_length=50) email = models.EmailField(max_length=75) def __unicode__(self): return self.name class Entry(models.Model): blog = models.ForeignKey(Blog) headline = models.CharField(max_length=255) body_text = models.TextField() pub_date = models.DateTimeField(auto_now_add=True) authors = models.ManyToManyField(Author) n_comments = models.IntegerField() n_pingbacks = models.IntegerField() rating = models.IntegerField() def __unicode__(self): return self.headline class BookManager(models.Manager): def create_book(title): book = self.create(title=title) return book class Book(models.Model): title = models.CharField(max_length=100) author = models.ForeignKey(Author) price_and_reviews = models.DecimalField(max_digits=8, decimal_places=2) from django.core.signals import post_save from django.dispatch import receiver
```

# Fernando Espíndola

Twitter @feresp

Github @fernandoe



# What is Django?

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design

Django makes it easier to build better web apps more quickly and with less code.

**The web framework for perfectionists with deadlines**

KISS Principle



# Who uses Django?

- ✦ HP
- ✦ Disqus
- ✦ Instagram
- ✦ Pinterest
- ✦ Mozilla
- ✦ Rdio
- ✦ Open Stack
- ✦ Globo
- ✦ Bitbucket
- ✦ Prezi
- ✦ DropBox
- ✦ Spotify
- ✦ Nasa
- ✦ The Washington Post
- ✦ The Guardian
- ✦ National Geographic



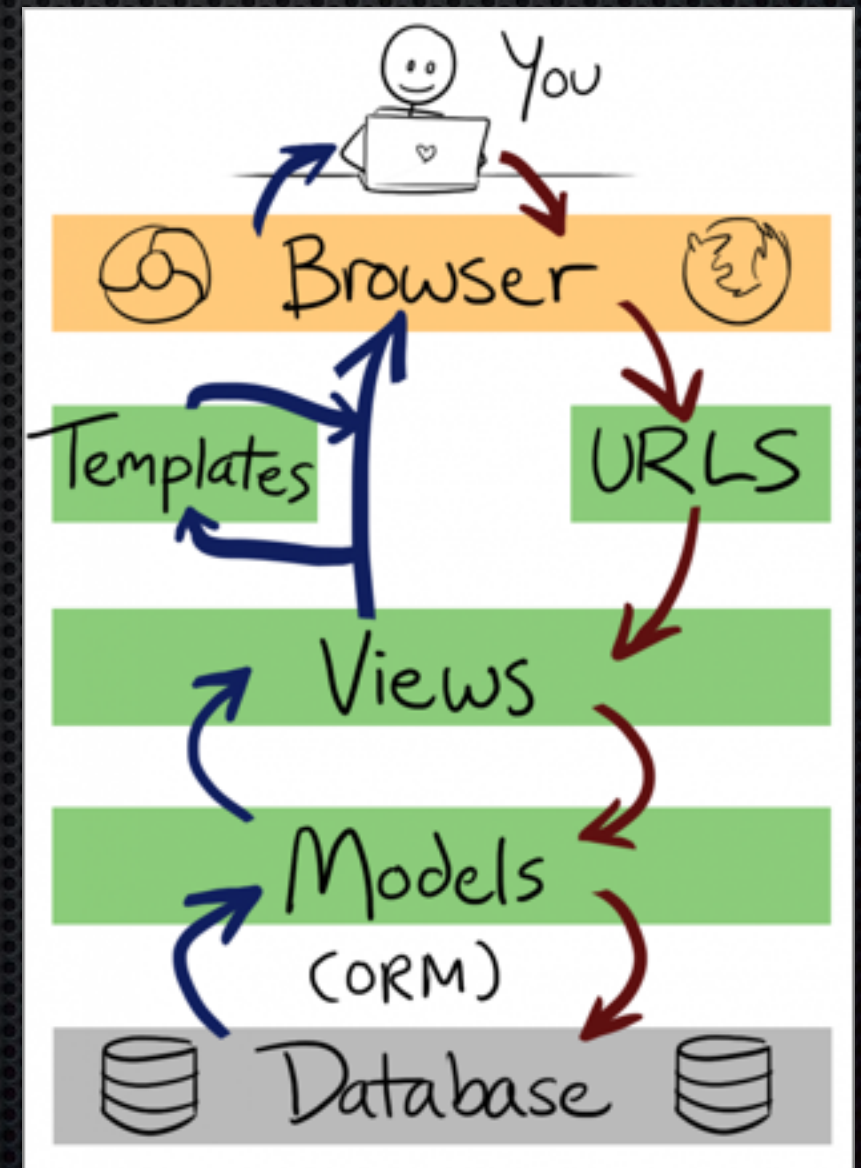
# Batteries Included

- ✦ ORM
- ✦ Migrations
- ✦ Templates
- ✦ Admin Site
- ✦ Authentication/  
Authorization
- ✦ Forms
- ✦ Validators
- ✦ Caching
- ✦ Comments
- ✦ GeoDjango
- ✦ I18N
- ✦ CSRF protection
- ✦ Sites
- ✦ Testing
- ✦ Dev Server
- ✦ Elegant URL Design



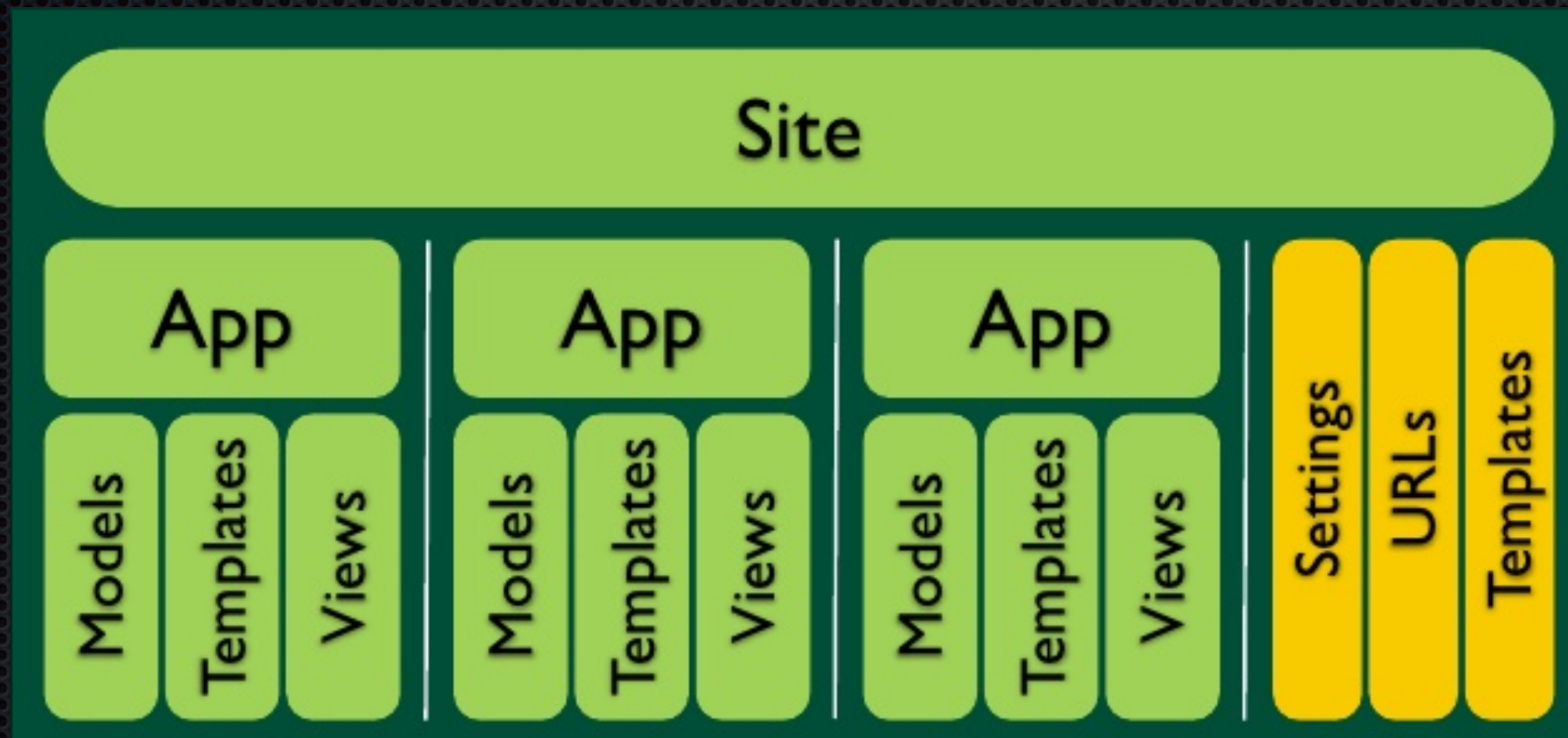
# Django is a framework

- ✦ Model (app/models.py)
- ✦ Template (app/templates/\*.html)
- ✦ View (app/views.py)



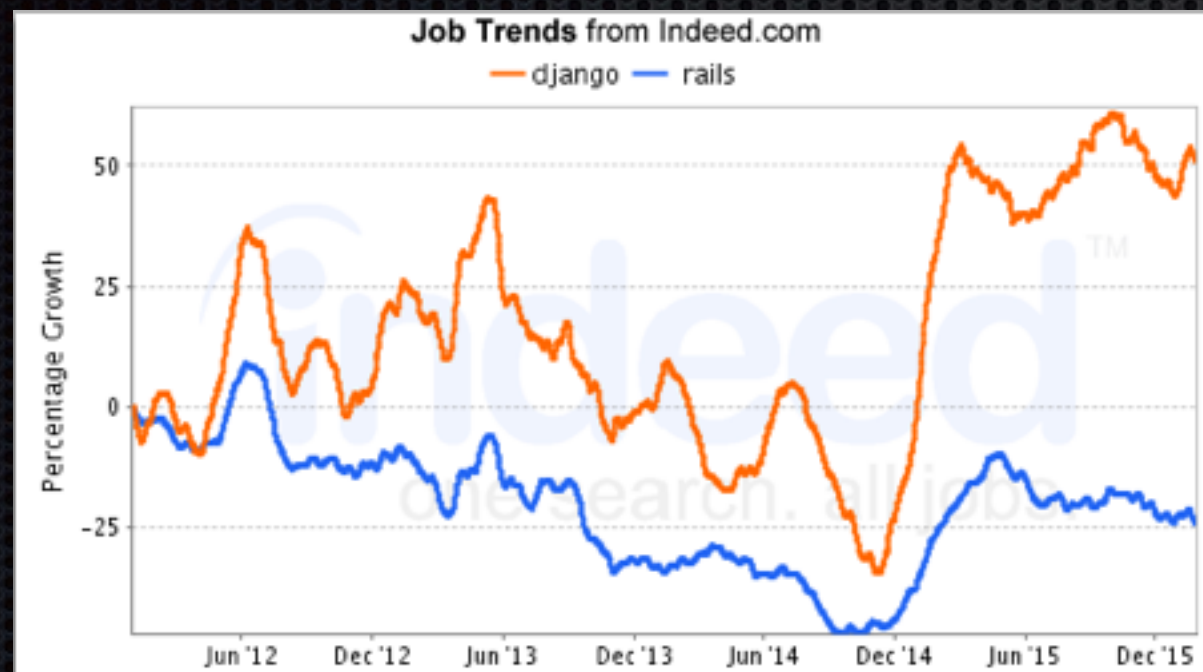


# Django Site Structure





# Job Trends



<http://www.indeed.com/jobtrends?q=+django%2C+rails&relative=1>



<http://www.indeed.com/jobtrends?q=+django%2C+php&relative=1>



# Hands-On



**Django Tutorial**

<https://goo.gl/YtfU2z>

