# DEVELOPING REUSABLE DJANGO APPS

PyCon Ukraine 2010

### Apps

```
INSTALLED_APPS = (
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.sites',
   'django.contrib.admin',
   'cms',
   'contact',
```

django.contrib.auth django.contrib.admin



## Advantages

- Improve Quality and Flexibility
- ▶ Great Ecosystem
- ▶ Functionality for free
- ▶ Reuse over and over
- ▶ Cut down on time and effort
- ▶ Head start on new Projects

### Non-Reusable Apps

- Feature creep
- ▶ Whole site is in one app
- Not going to reuse

#### **Focus**

What does this application do?

#### **Good Focus**

"Handle entries in a blog"

#### **Bad Focus**

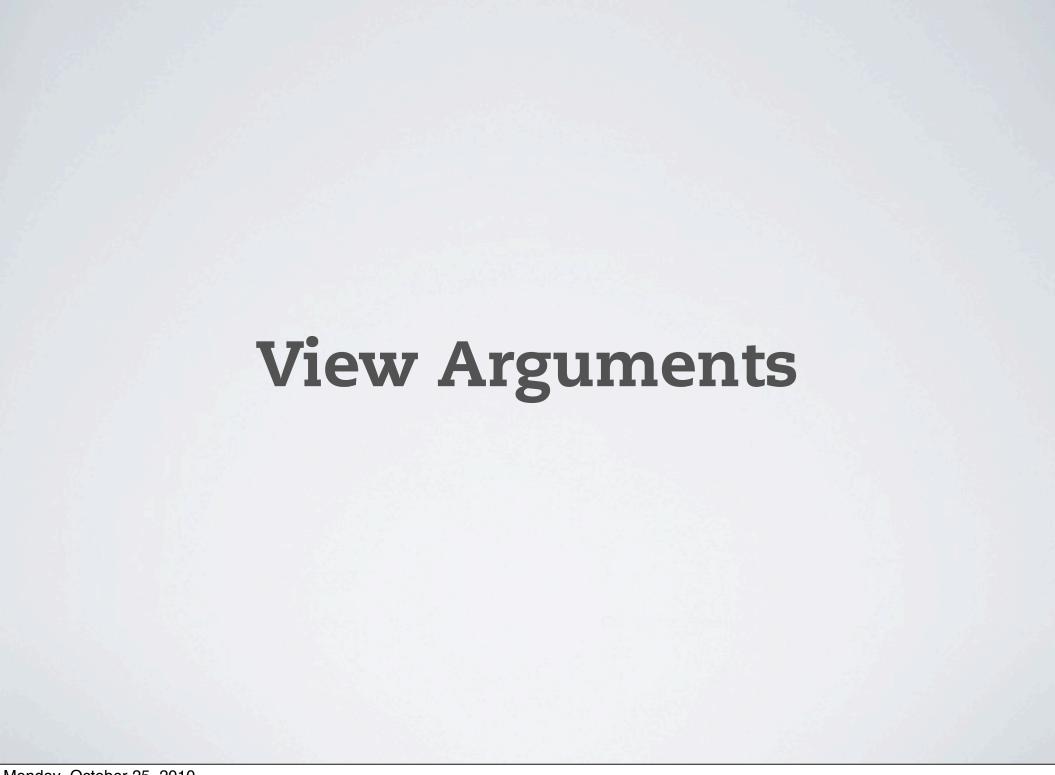
"Handle entries in a blog, and users who post them, and their authentication, and tagging, and categorization ..."

# Orthogonality

Changing a feature doesn't affect the other one

#### Guidelines

- Unrelated Features
- Orthogonality
- ▶ Reuse





from django import forms

```
class ContactForm(forms.Form):
    subject = forms.CharField(max_length=100)
    message = forms.CharField()
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)
```

```
def contact(request):
  if request.method == 'POST':
     form = ContactForm(request.POST)
     if form.is_valid():
        # Process the data in form.cleaned_data
        return HttpResponseRedirect('/thanks/')
  else:
     form = ContactForm()
  return render_to_response('contact.html', {
     'form': form,
```

```
def contact(request, form_class=ContactForm):
  if request.method == 'POST':
     form = form_class(request.POST)
     if form.is_valid():
        # Process the data in form.cleaned data
        return HttpResponseRedirect('/thanks/')
  else:
     form = form_class()
  return render_to_response('contact.html', {
     'form': form,
```

```
(r'contact/',
  'contactform.views.contact', {
     'form_class': MyContactForm,
```

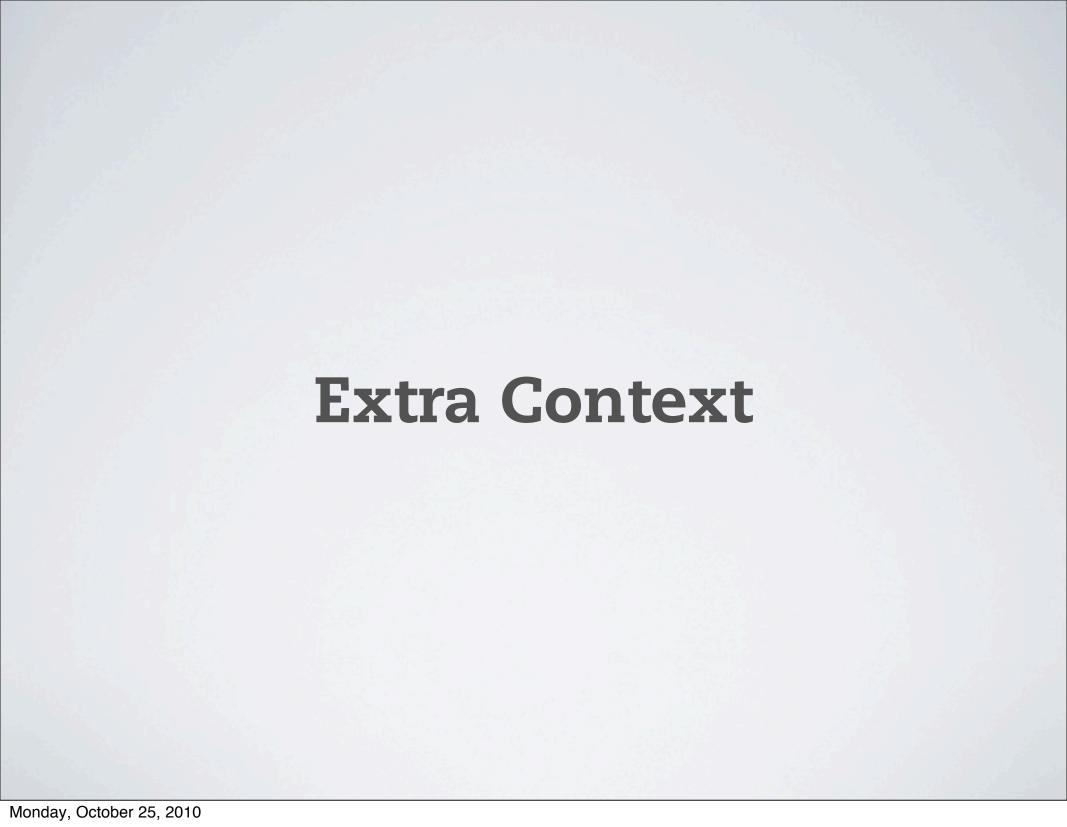


```
def contact(request, form_class=ContactForm,
           template_name='contact.html'):
  if request.method == 'POST':
     form = form_class(request.POST)
     if form.is_valid():
       # Process the data in form.cleaned data
       return HttpResponseRedirect('/thanks/')
  else:
     form = form_class()
  return render_to_response(template_name, {
     'form': form,
```

```
(r'contact/',
  'contactform.views.contact', {
     'form_class': MyContactForm,
     'template_name': 'contact/mycontactform.html',
```



```
def contact(request, form_class=ContactForm,
           template_name='contact.html',
           success_url='/thanks/'):
  if request.method == 'POST':
     form = form_class(request.POST)
     if form.is_valid():
        # Process the data in form.cleaned data
        return HttpResponseRedirect(success_url)
  else:
     form = form_class()
  return render_to_response(template_name, {
     'form': form,
```



```
def contact(request, ..., extra_context={}):
   if extra_context is None: extra_context = {}
  if request.method == 'POST':
  else:
     form = form_class()
t = template_loader.get_template(template_name)
c = RequestContext(request, {
  'form': form,
}, context_processors)
for key, value in extra_context.items():
   if callable(value):
      c[key] = value()
   else:
      c[key] = value
return HttpResponse(t.render(c))
```

```
(r'contact/',
    'contactform.views.contact', {
        'form_class': MyContactForm,
        'template_name': 'contact/mycontactform.html',
        'success_url': '/',
        'extra_context': {'foo': 'bar'},
})
```

# Flexible URL Handling

```
urlpatterns = patterns(",
    url(r'^$', 'contact.views.contact', name='contact'),
)
```

# Flexible URL Handling

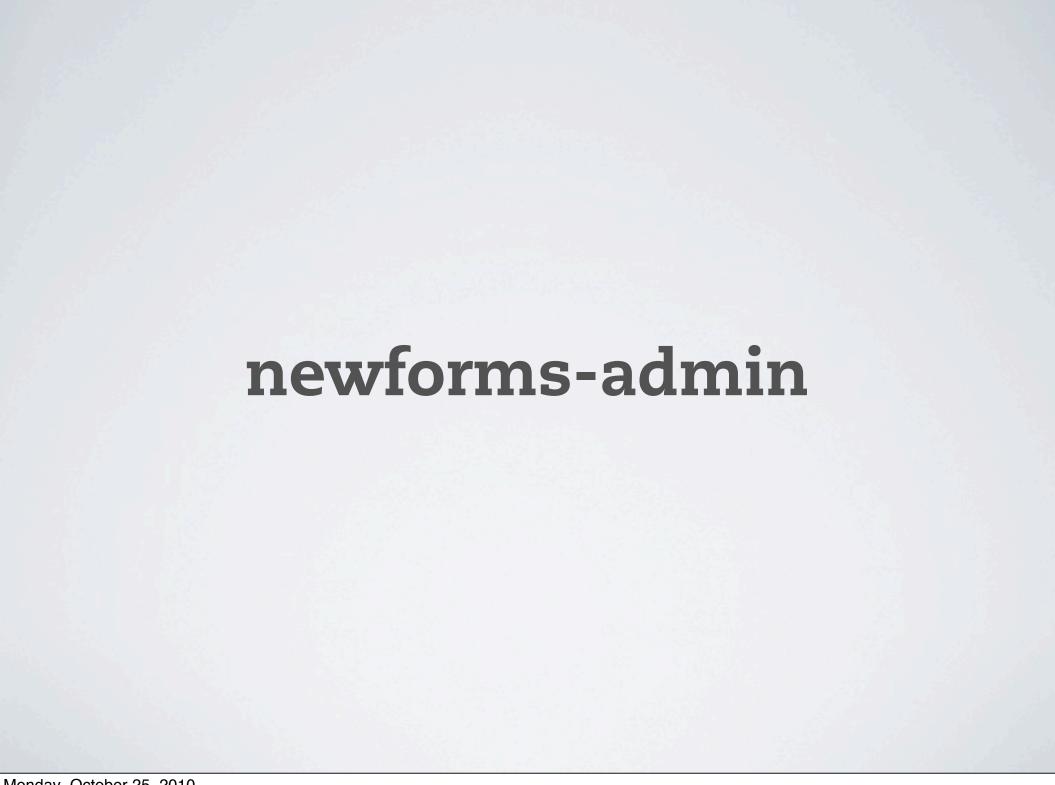
- Don't hardcode URLs
- django.db.models.permalink
  - {% url %}
- django.core.urlresolvers.reverse()

```
def post_detail(request, pk):
    post = get_object_or_404(Post, pk=pk)
    return render_to_response('post_detail.html', {
        'post': post
    })
```

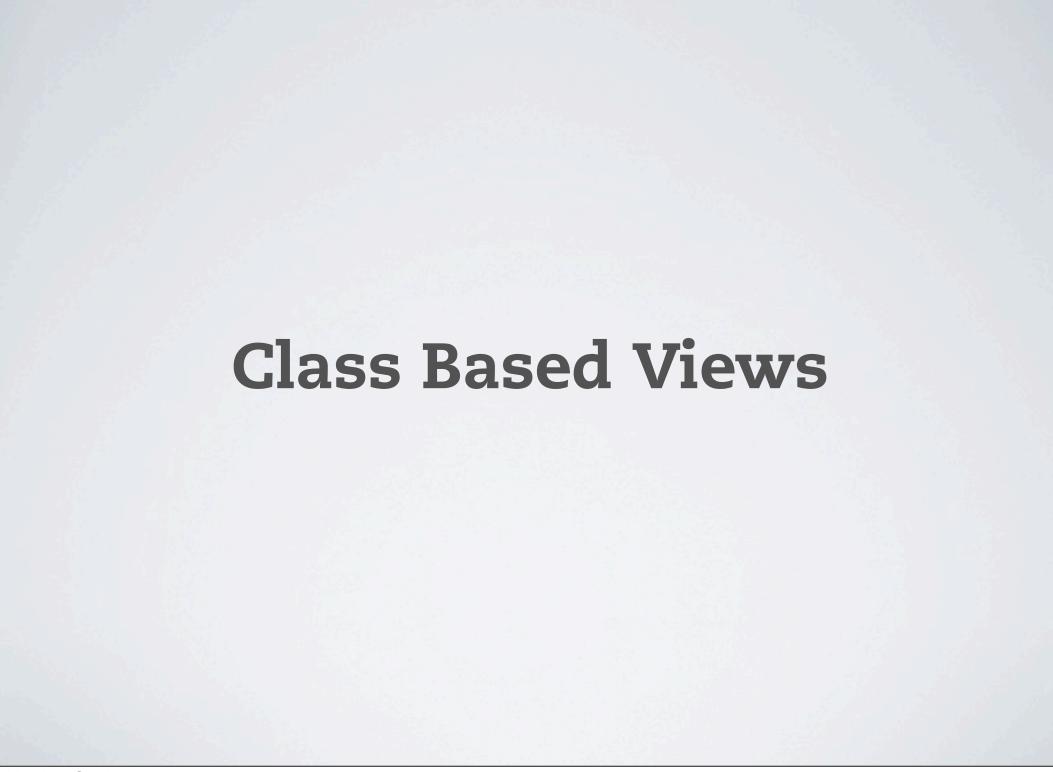
```
(r'post/(\d+)/',
  'django.views.generic.list_detail.object_detail', {
     'queryset': Post.objects.all()
```

#### **Generic Views**

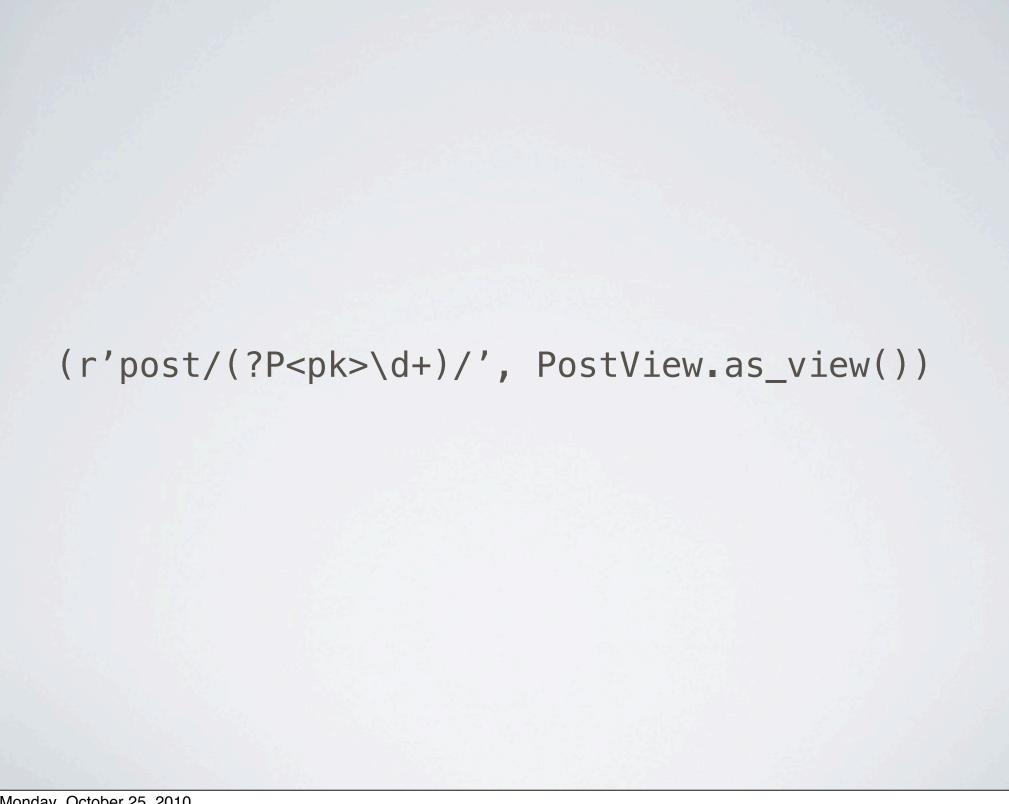
def object\_detail(request, year, month, day, queryset, date\_field, month\_format='%b', day\_format='%d', object\_id=None, slug=None, slug\_field='slug', template\_name=None, template\_name\_field=None, template\_loader=loader, extra\_context=None, context\_processors=None, template\_object\_name='object', mimetype=None, allow\_future=False):

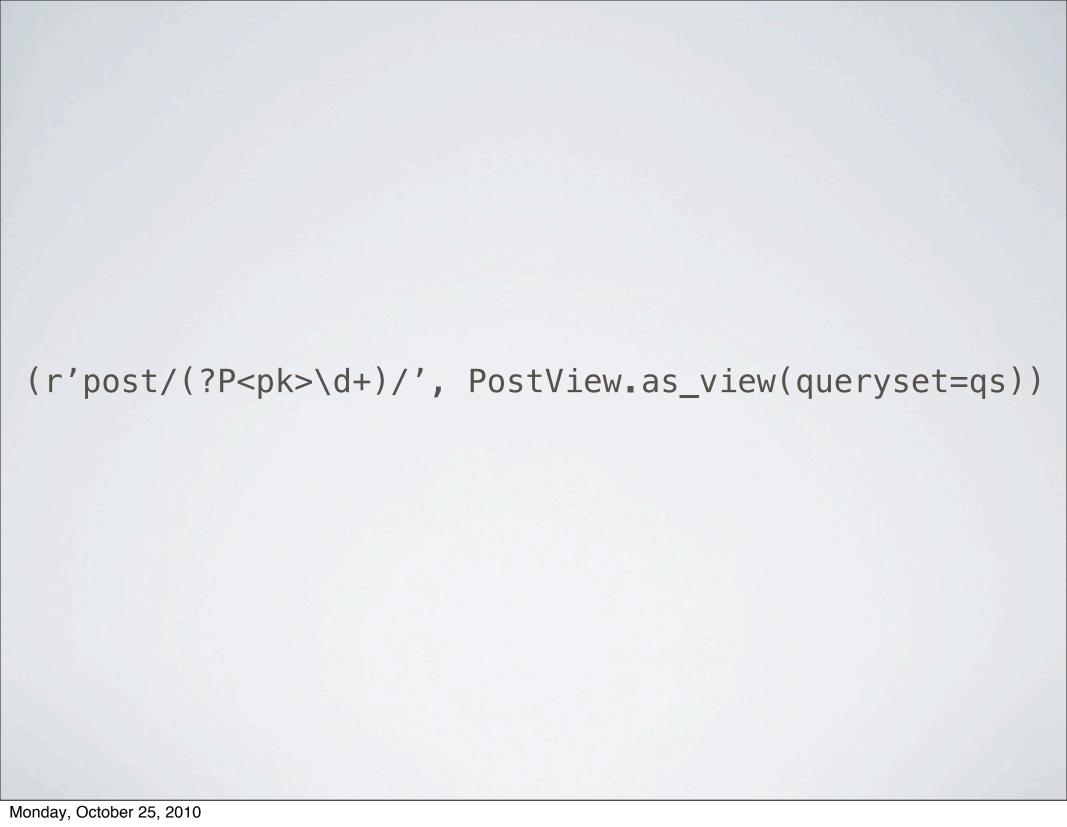


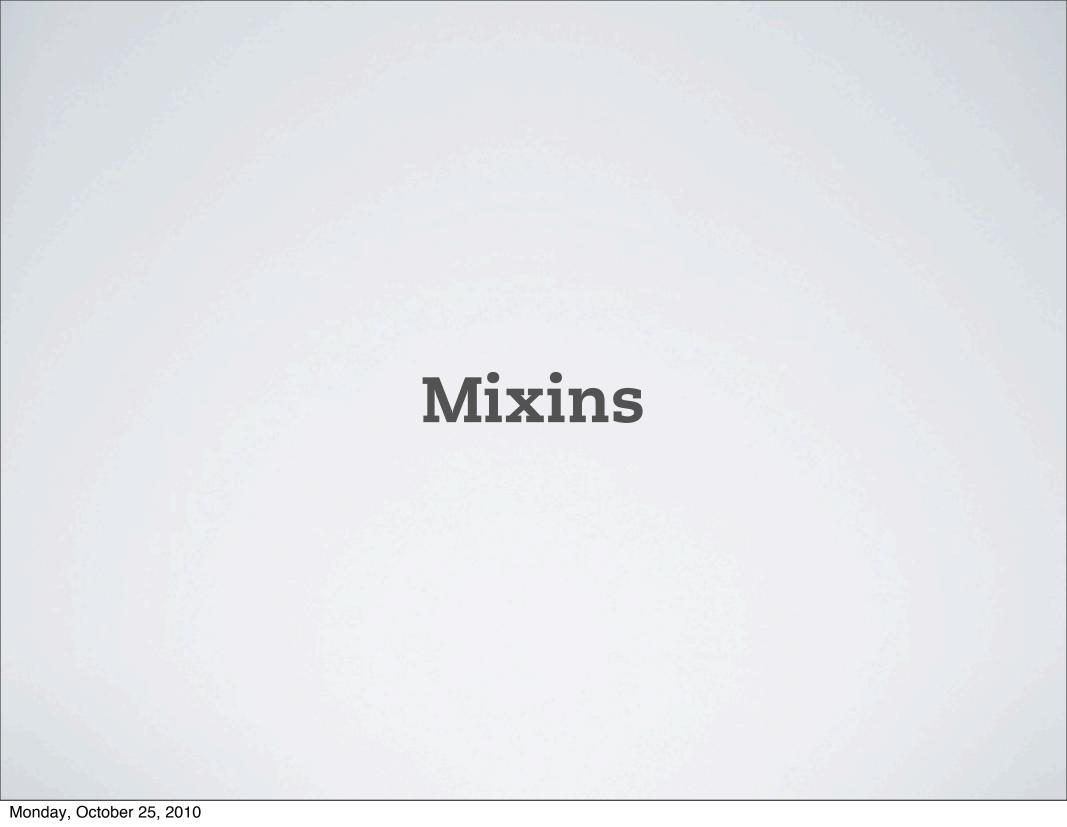
```
class BookAdmin(admin.ModelAdmin):
  def save_model(self, request, obj, form, change):
    if not change: obj.author = request.user
    obj.save()
  def has_change_permission(self, request, obj=None):
    if not obj:
      return True
    if obj.author == request.user:
      return True
    else:
      return False
admin.site.register(Book, BookAdmin)
```



class PostView(DetailView):
 queryset = Post.objects.all()

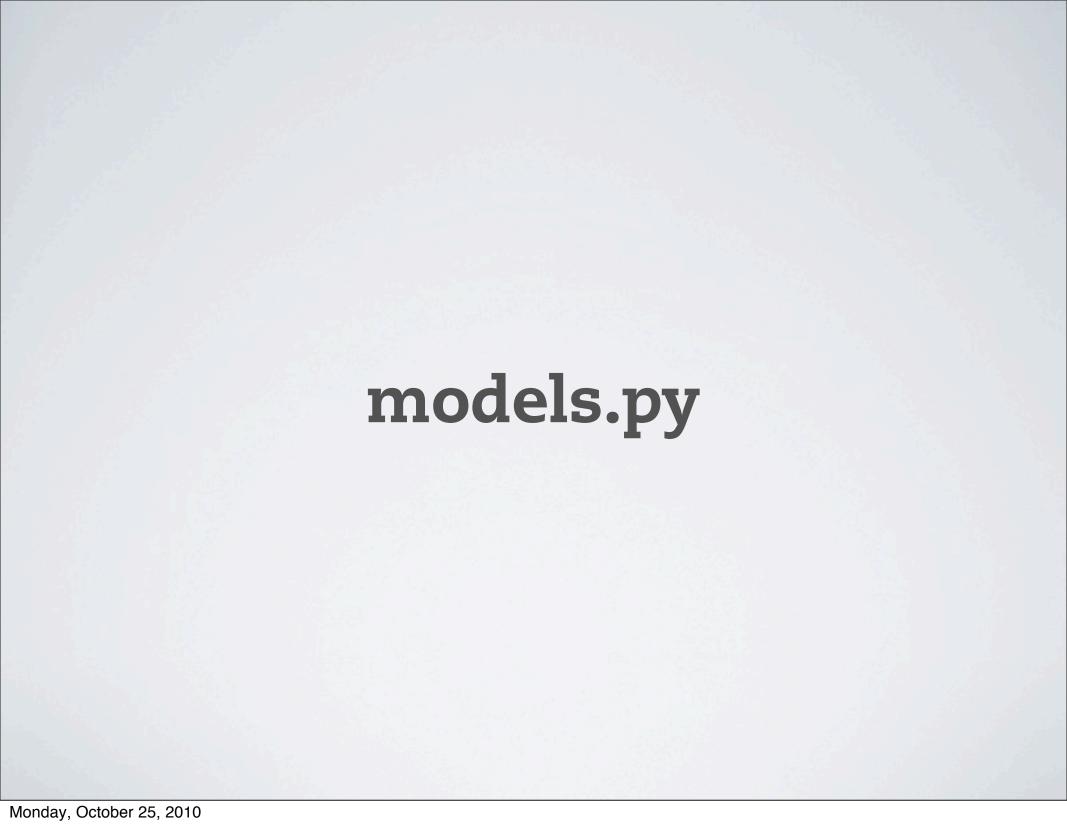






class PostView(JinjaMixin, DetailView):
 queryset = Post.objects.all()





```
APP_CLASSES = (
    'myauth.MyAuthApp',
)
```

## apps.py

from django.contrib.auth import AuthApp

class MyAuthApp(AuthApp): verbose\_name = 'My Custom Authentication App'





from django.contrib.auth import AuthApp

class MyAuthApp(AuthApp):

def models(self):
 return self.models.iteritems()
models = property(models)

## Questions?

@arthurk