REVSYS | revolution systems

Introduction to Django

Jacob Kaplan-Moss

Strange Loop 2011

http://lanyrd.com/sfypm

Django Training

Part 1: Introduction to Django.

“*Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.*”

“ *Django is a **high-level Python web framework** that encourages rapid development and clean, pragmatic design.* ”

“*Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.*”

“*Django is a high-level Python web framework that encourages rapid development* ***and clean, pragmatic design.***”

Documentation

http://django.me/design

http://www.djangobook.com/en/2.0/chapter01/

Which Django version should I use?

Installing Django

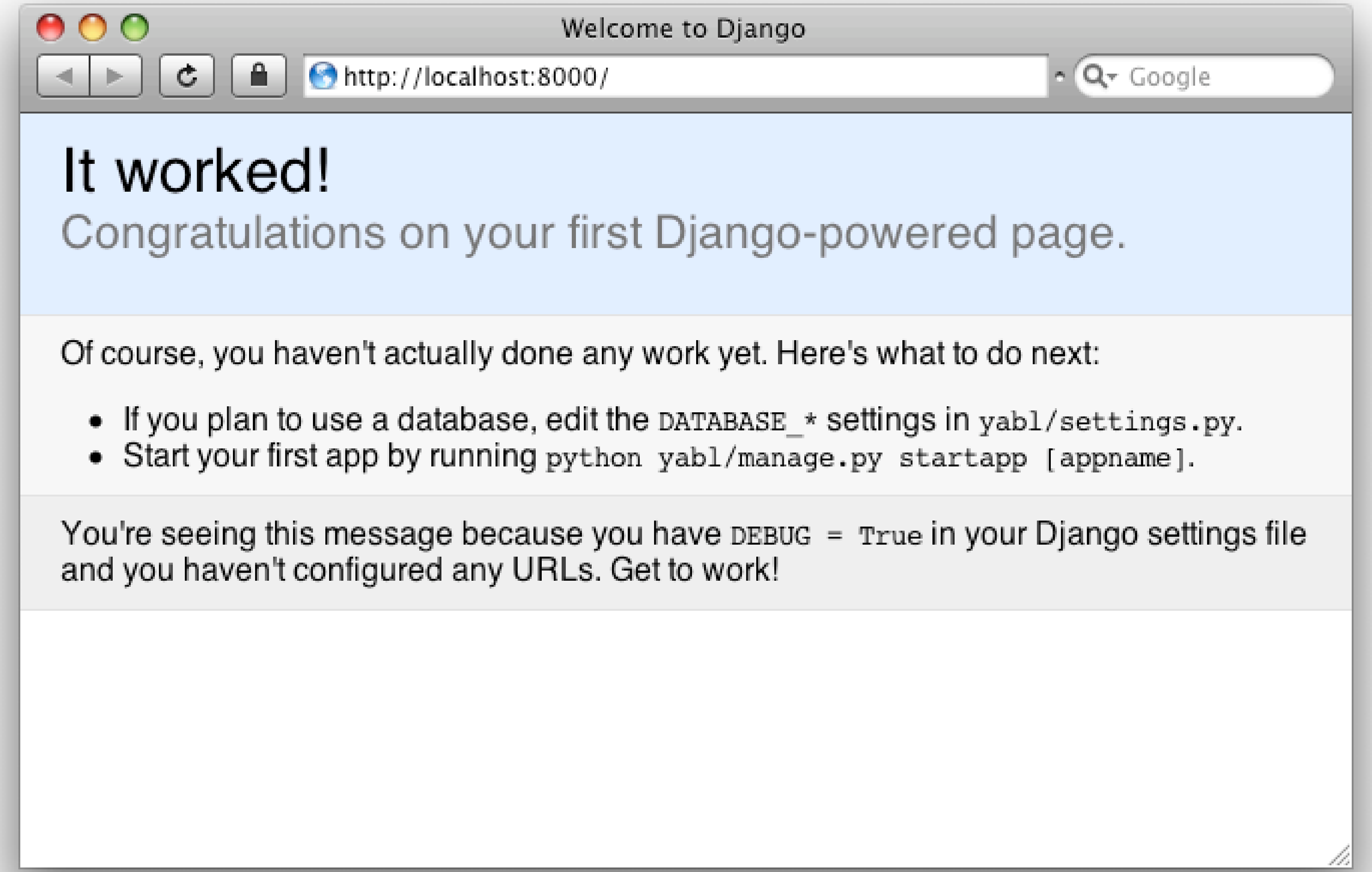
- Download and run **`http://bit.ly/dsetup`**
  (`http://python-distribute.org/distribute_setup.py`)
- **`easy_install Django`**
- Later: learn about `pip` and `virtualenv`
  `http://pip.rtfd.org/`
  `http://virtualenv.rtfd.org/`

“Projects”

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

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

```
$ python manage.py runserver
Validating models...
0 errors found.

Django version 1.1 beta 1 SVN-10844, using settings 'yabl.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



It worked!

Congratulations on your first Django-powered page.

Of course, you haven't actually done any work yet. Here's what to do next:

- If you plan to use a database, edit the `DATABASE_*` settings in `yabl/settings.py`.
- Start your first app by running `python yabl/manage.py startapp [appname]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Project settings

- DATABASE_ENGINE
- DATABASE_NAME
- DATABASE_USER
- DATABASE_PASSWORD
- DATABASE_HOST

```
$ python manage.py syncdb
Creating table auth_permission
Creating table auth_group
Creating table auth_user
Creating table auth_message
Creating table django_content_type
Creating table django_session
Creating table django_site

You just installed Django's auth system,
which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username (Leave blank to use 'jacob'): jacob
E-mail address: jacob@jacobian.org
Password:
Password (again):
Superuser created successfully.
Installing index for auth.Permission model
Installing index for auth.Message model
```

Documentation

http://django.me/about-settings

http://django.me/settings

http://django.me/manage.py

Exercise:

“it worked!”

REVSYS | revolution systems

Django Training

Part 2: Apps, models, and the admin

“Apps”

“Models”

What’s a model?

MVC?

(Model-View-Controller)

```
CREATE TABLE "entries_entry" (
    "id" integer NOT NULL PRIMARY KEY,
    "author_id" integer NOT NULL,
    "pub_date" datetime NOT NULL,
    "headline" varchar(200) NOT NULL,
    "slug" varchar(50) NOT NULL UNIQUE,
    "summary" text NOT NULL,
    "body" text NOT NULL
)
```

Scary Quirky Language

- SQL is tough
- SQL knows no version control
- DRY
- Python is fun!

```
import datetime
from django.db import models
from yabl.authors.models import Author

class Entry(models.Model):
    author        = models.ForeignKey(Author, related_name='entries')
    pub_date      = models.DateTimeField(default=datetime.datetime.now)
    headline      = models.CharField(max_length=200)
    slug          = models.SlugField(unique=True)
    summary       = models.TextField()
    body          = models.TextField()
```

Defining Models

```
$ python manage.py startapp authors
```

```
authors/
   __init__.py
   models.py
   tests.py
   views.py
```

```
  INSTALLED_APPS = (
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.sites",
    "yabl.authors",
)
```

```
from django.db import models

class Author(models.Model):
    first_name  = models.CharField(max_length=200)
    last_name   = models.CharField(max_length=200)
    bio         = models.TextField(blank=True)
```

```
$ python manage.py validate
0 errors found.
```

```
$ python manage.py sqlall authors
BEGIN;
CREATE TABLE "authors_author" (
    "id" integer NOT NULL PRIMARY KEY,
    "first_name" varchar(200) NOT NULL,
    "last_name" varchar(200) NOT NULL,
    "bio" text NOT NULL
);
COMMIT;
```

```
$ python manage.py syncdb
Creating table authors_author
Installing index for authors.Author model
```

```
$ python manage.py shell

[1] >>> from yabl.authors.models import Author

[2] >>> a = Author(first_name="John", last_name="Barth")

[3] >>> a.save()
```

```
[4] >>> Author.objects.all()
[4]   : [<Author: Author object>]

[5] >>> Author.objects.create(first_name='Miguel', last_name='de Cervantes')
[5]   : <Author: Author object>

[6] >>> Author.objects.all()
[6]   : [<Author: Author object>, <Author: Author object>]

[7] >>> al = Author.objects.filter(first_name='John')

[8] >>> al[0].last_name
[8]   : u'Barth'

[9] >>> Author.objects.get(last_name__startswith='de').first_name
[9]   : u'Miguel'
```

Model metadata

```
class Author(models.Model):
    first_name  = models.CharField(max_length=200)
    last_name   = models.CharField(max_length=200)
    bio         = models.TextField(blank=True)

    def __unicode__(self):
        return '%s %s' % (self.first_name, self.last_name)
```

```
class Author(models.Model):
    …

    class Meta:
        verbose_name_plural = 'authors'
        ordering = ['last_name', 'first_name']
```

```
[1] >>> from yabl.authors.models import Author

[2] >>> Author.objects.all()
[2]   : [<Author: John Barth>, <Author: Miguel de Cervantes>]

[3] >>> Author.objects.order_by('-first_name')
[3]   : [<Author: Miguel de Cervantes>, <Author: John Barth>]
```

Documentation

http://django.me/models

http://www.djangobook.com/en/2.0/chapter05/

Exercise:

Write some apps and some models:

- Author (`authors` app)
    - first_name (CharField)
    - last_name (CharField)
    - bio (TextField)
- Entry (`entries` app)
    - author (ForeignKey)
    - pub_date (DateTimeField)
    - is_published (BooleanField)
    - headline (CharField)
    - slug (SlugField)
    - summary (TextField)
    - body (TextField)

Django’s admin interface

“*A Web-based interface, limited to trusted site administrators, that enables the adding, editing and deletion of site content.*”

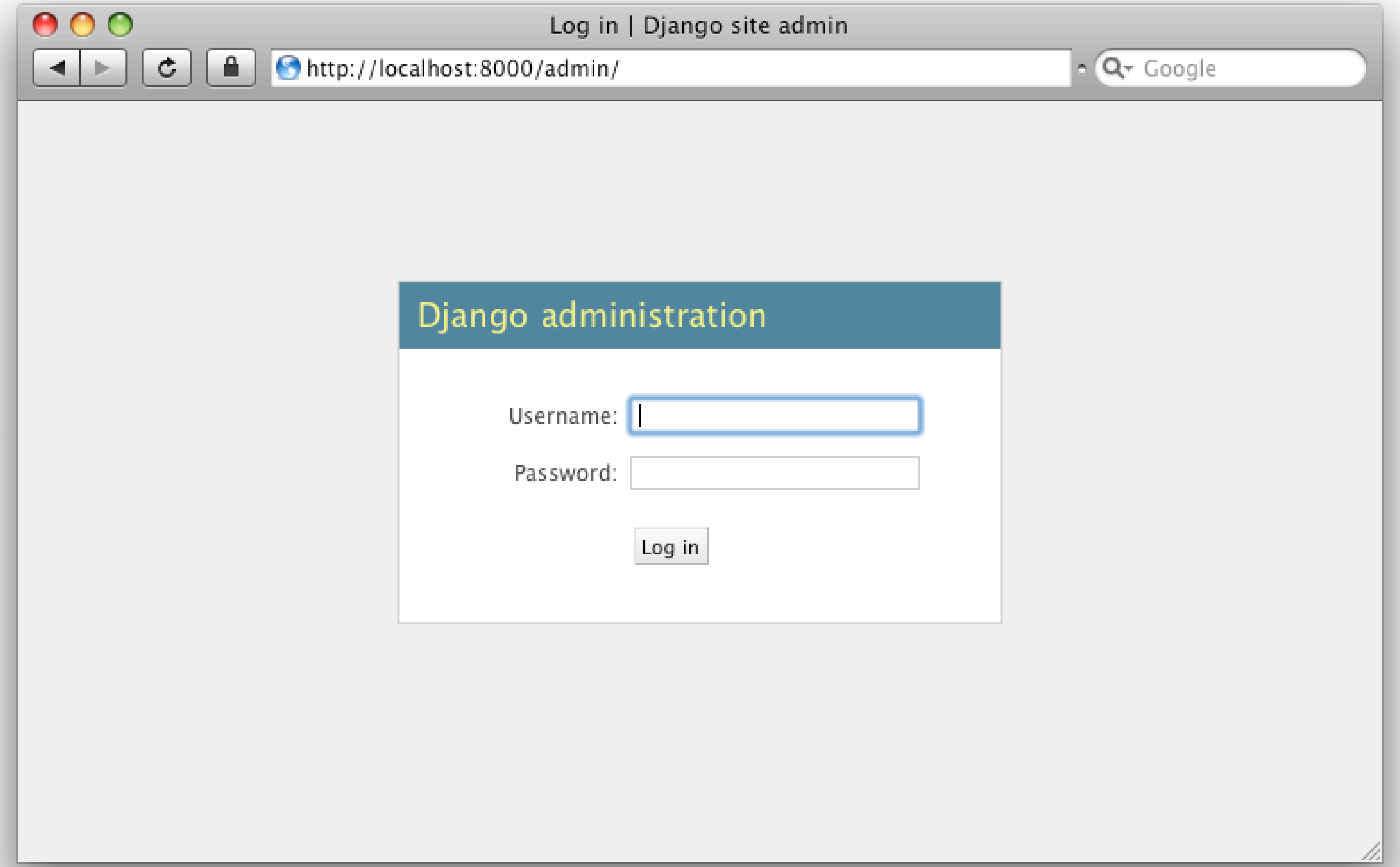
— The Django Book
http://djangobook.com/en/2.0/chapter06/

```
from django.contrib import admin
from yabl.authors.models import Author

admin.site.register(Author)
```

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.admin',
    'yabl.authors',
    'yabl.entries',
)
```

```
$ python manage.py syncdb
Creating table django_admin_log
Installing index for admin.LogEntry model
```

```
from django.conf.urls.defaults import *

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

urlpatterns = patterns('',
    # Example:
    # (r'^yabl/', include('yabl.foo.urls')),

    # Uncomment the admin/doc line below and add 'django.contrib.admindocs'
    # to INSTALLED_APPS to enable admin documentation:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Uncomment the next line to enable the admin:
    (r'^admin/', include(admin.site.urls)),
)
```

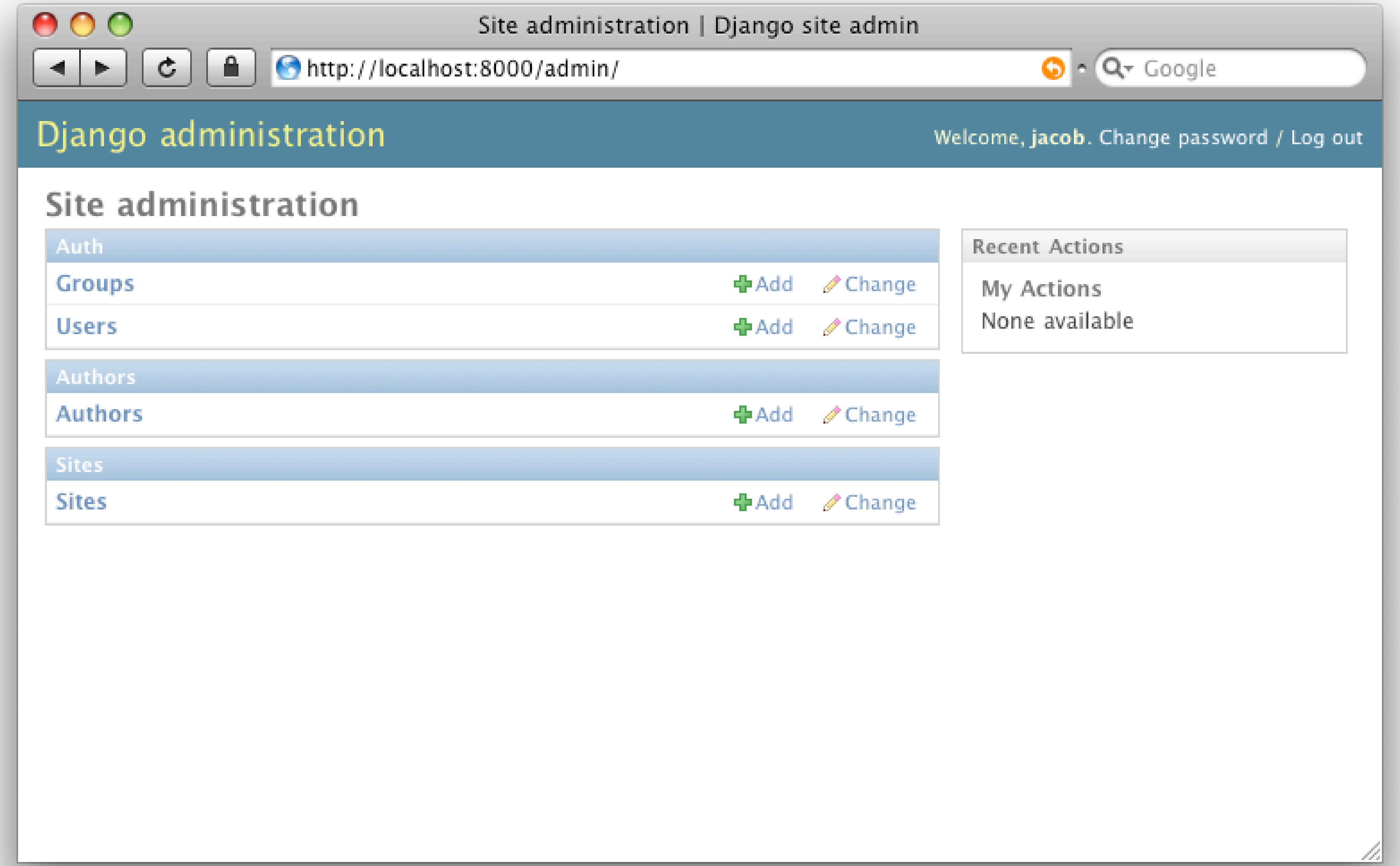

Log in | Django site admin

http://localhost:8000/admin/

Django administration

Username:

Password:



Site administration | Django site admin

http://localhost:8000/admin/

Django administration

Welcome, **jacob**. Change password / Log out

Site administration

Auth		
Groups	Add	Change
Users	Add	Change

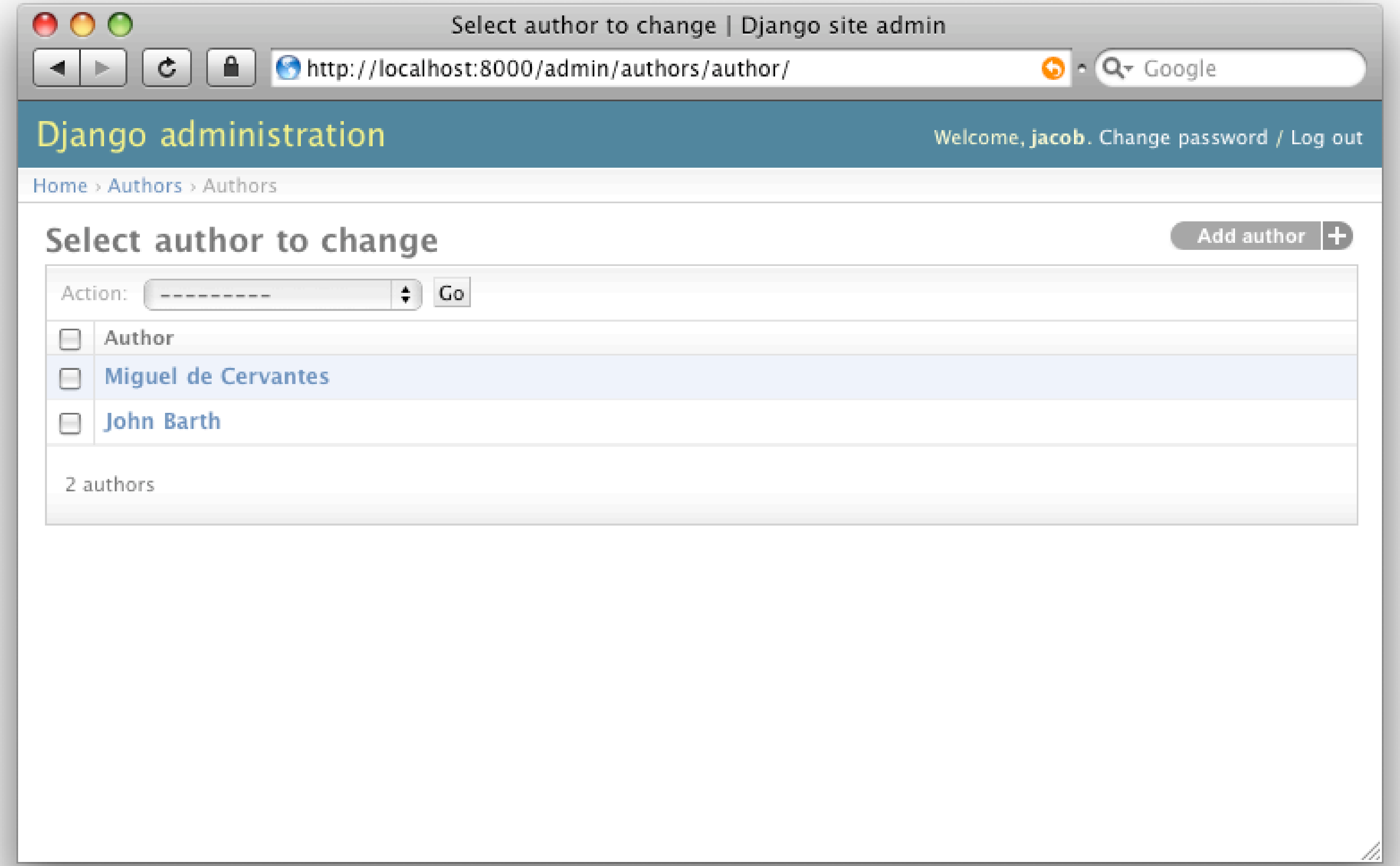
Authors		
Authors	Add	Change

Sites		
Sites	Add	Change

Recent Actions

My Actions

None available


Select author to change | Django site admin

http://localhost:8000/admin/authors/author/

Django administration

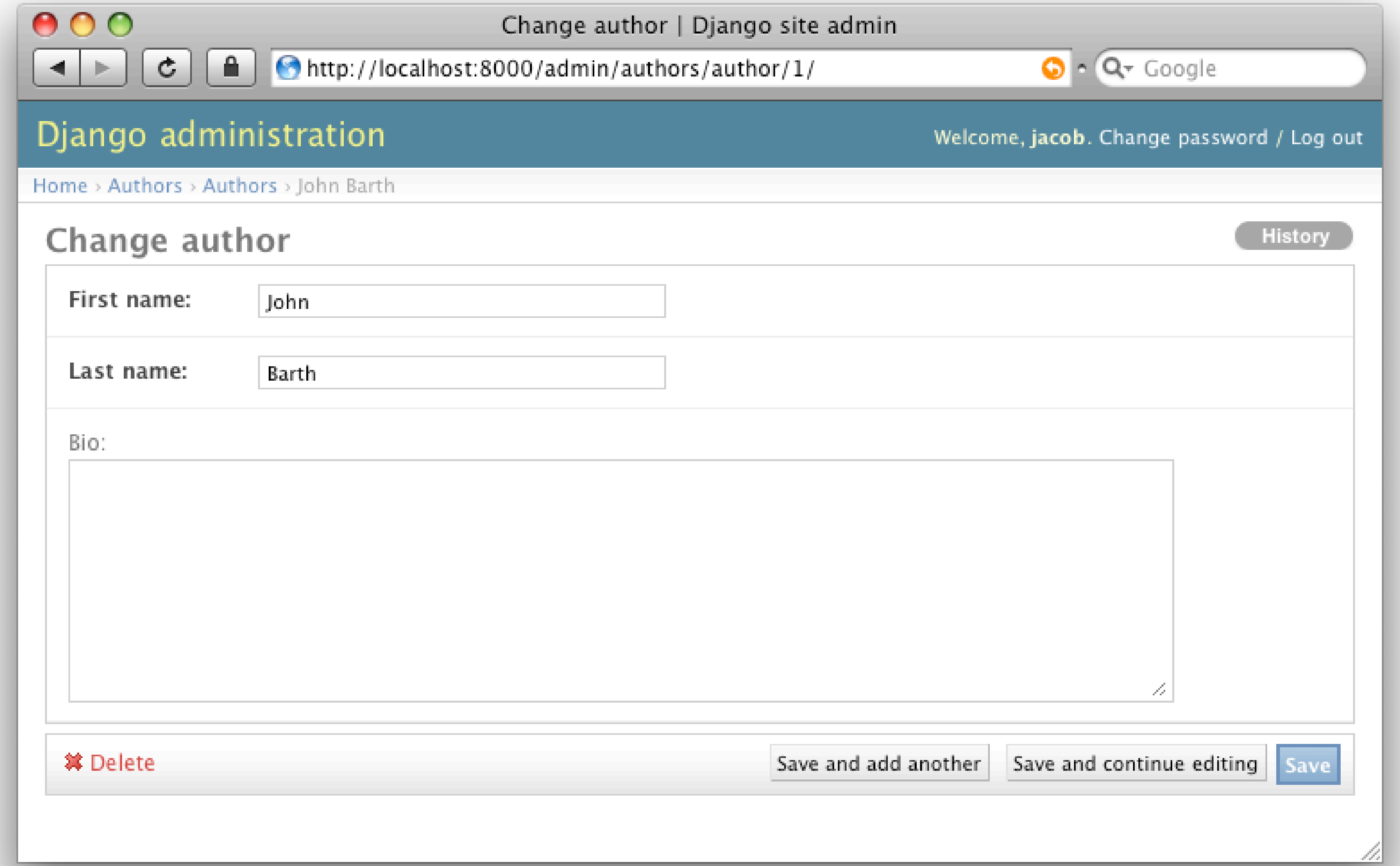
Welcome, **jacob**. Change password / Log out

Home › Authors › Authors

Select author to change

	Author
☐	Miguel de Cervantes
☐	John Barth

2 authors



Change author | Django site admin

http://localhost:8000/admin/authors/author/1/

Django administration

Welcome, **jacob**. Change password / Log out

Home › Authors › Authors › John Barth

Change author

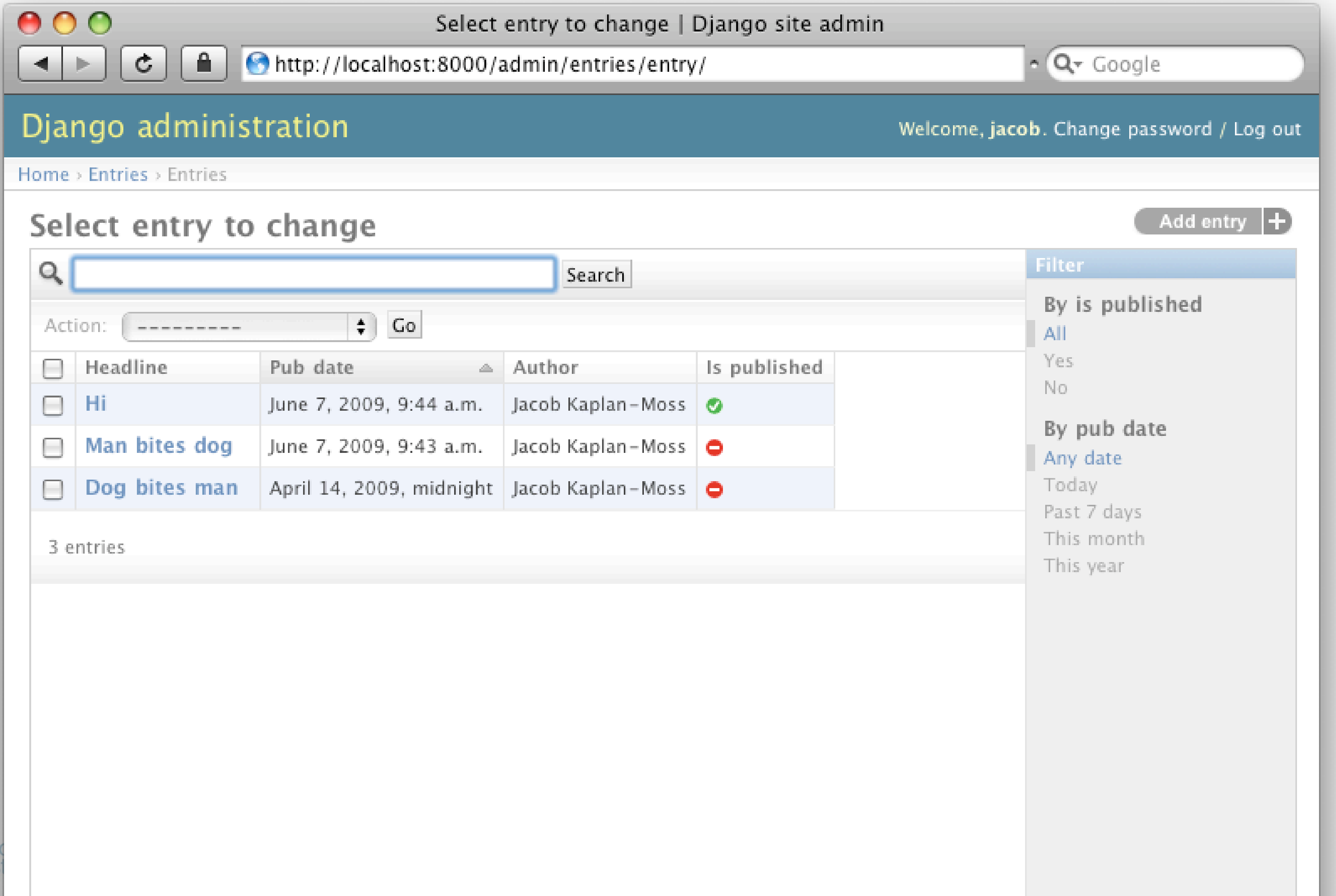
First name:	John
Last name:	Barth
Bio:	

```
from django.contrib import admin
from yabl.authors.models import Author

class AuthorAdmin(admin.ModelAdmin):
    pass

admin.site.register(Author, AuthorAdmin)
```

Documentation

http://djangobook.com/en/2.0/chapter06/
http://django.me/admin

Exercise:



Select entry to change | Django site admin

http://localhost:8000/admin/entries/entry/

Django administration

Welcome, **jacob**. Change password / Log out

Home › Entries › Entries

Select entry to change

Headline	Pub date	Author	Is published
Hi	June 7, 2009, 9:44 a.m.	Jacob Kaplan-Moss	✓
Man bites dog	June 7, 2009, 9:43 a.m.	Jacob Kaplan-Moss	⊖
Dog bites man	April 14, 2009, midnight	Jacob Kaplan-Moss	⊖

3 entries

By is published
- All
- Yes
- No

By pub date
- Any date
- Today
- Past 7 days
- This month
- This year

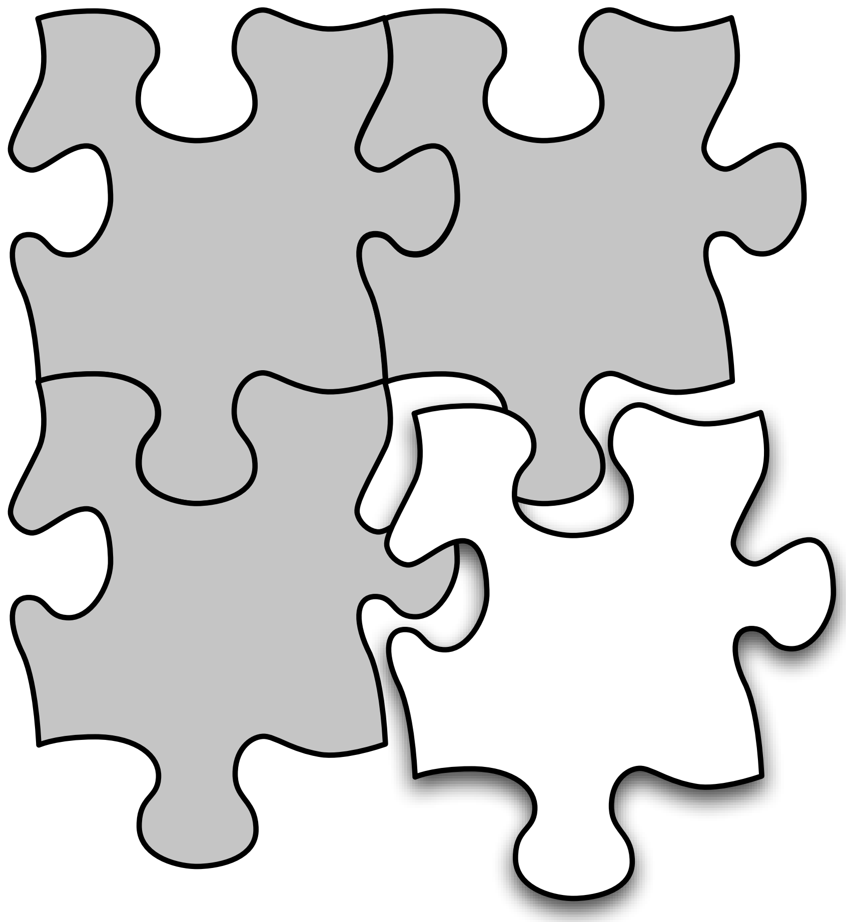
REVSYS | revolution systems

Django Training

Part 3: URLs, views, and templates

Views

What’s a view?



URLs

page.php

script.cgi?pageid=144

StoryPage.aspx

0,2097,1-1-30-72-407-4752,00.html

/authors/
/authors/jacob/
/authors/adrian/

```
ROOT_URLCONF = "yabl.urls"
```

yabl/urls.py

```
from django.conf.urls.defaults import *

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

urlpatterns = patterns('',
    (r'^authors/$',         'yabl.authors.views.author_list'),
    (r'^authors/(\d+)/$',   'yabl.authors.views.author_detail'),
    (r'^admin/', include(admin.site.urls)),
)
```

yabl/authors/urls.py

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^$',          'yabl.authors.views.author_list'),
    (r'^(\d+)/$',    'yabl.authors.views.author_detail'),
)
```

yabl/urls.py

```
from django.conf.urls.defaults import *

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

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

Regex crash course

`a`	The **letter “a”**.
`a+`	**One or more** “a”s.
`b?`	**Zero or one** “b”s.
`c{1,3}`	**One, two, or three** “c”s.
`.`	**Any single character**.
`[abc]`	**Either** an “a”, “b”, or “c”.
`[A-Z]`	Any character **between “A” and “Z”**.
`[A-Za-z0-9]?`	Zero or one letters **“A-Z”, “a-z”, or “0-9”.**
`(\d{3,4})`	**A group** containing three or four digits.
`(\w*)`	A group containing zero or more **word characters** (letters/digits).
`[^/]+`	One or more characters **until (and not including) a forward slash**.
`^(joe|bob)`	A string **starting with** “joe” **or** “bob”.
`(?P<id>\d+)`	A **group named “id”** containing one or more digits.
`article/$`	A string **ending with** “article/”

REVSYS revolution systems

Dissecting a request

- `GET /authors/1/`
- `ROOT_URLCONF`
- `yabl.urls`
- `(r'^authors/', include('yabl.authors.urls'))`
- `yabl.authors.urls`
- `(r'^$', 'author_list')` *(no match)*
- `(r'^(\d+)/', 'author_detail')` *(match!)*
- `author_detail(request, '1')`

Documentation

http://django.me/urls

A first view

yabl/authors/views.py

```
from django.http import HttpResponse

def author_list(request):
    return HttpResponse("This is the author list!")
```

yabl/authors/views.py

```
from django.http import HttpResponse
from yabl.authors.models import Author

def author_list(request):
    r = "<ul>"
    for a in Author.objects.all():
        r += "<li>%s</li>" % a.name
    r += "</ul>"
    return HttpResponse(r)
```

yabl/authors/views.py

```
from django import template
from django.http import HttpResponse
from yabl.authors.models import Author

def author_list(request):
    as = Author.objects.all()
    tmpl = template.loader.get_template("authors/index.html")
    context = template.Context({"authors": as})
    return HttpResponse(tmpl.render(context))
```

yabl/authors/views.py

```
from django.shortcuts import render
from yabl.authors.models import Author

def author_list(request):
    context = {"authors" : Author.objects.all()}
    return render(request, "authors/index.html", context)
```

yabl/authors/views.py

```
from django.http import Http404
from django.shortcuts import render_to_response
from yabl.authors.models import Author

def author_detail(request, author_id):
    try:
        author = Author.objects.get(id=author_id)
    except Author.DoesNotExist:
        raise Http404()

    return render(request, "authors/detail.html", {"author" : author})
```

yabl/authors/views.py

```
from django.shortcuts import render_to_response, get_object_or_404
from yabl.authors.models import Author

def author_detail(request, author_id):
    author = get_object_or_404(Author, id=author_id)
    return render(request, "authors/detail.html", {"author" : author})
```

Templates

What’s a template?

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Authors</title></head>
<body>
   <h1>Authors ({{ authors|length }} total)</h1>
   <ul>
      {% for a in authors %}
         <li>
            <a href="{{ a.id }}/">{{ a.name }}</a>
         </li>
      {% endfor %}
   </ul>
</body>
</html>
```

Where to templates go?

- In an app's `templates` directory.
- In directories specified by `settings.TEMPLATE_DIRS`.
- ...

```
TEMPLATE_DIRS = [
    '/path/to/some/templates/',
    '/path/to/some/more/other/templates/',
]
```

```
TEMPLATE_DIRS = [
    '/Users/jacob/Projects/stl-django/yabl/templates/',
]
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Authors</title></head>
<body>
   <h1>Authors ({{ authors|length }} total)</h1>
   <ul>
      {% for a in authors %}
         <li>
            <a href="{{ a.id }}/">{{ a.name }}</a>
         </li>
      {% endfor %}
   </ul>
</body>
</html>
```

The magic dot

- a["name"]
- a.name
- a.name()

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Authors</title></head>
<body>
   <h1>Authors ({{ authors|length }} total)</h1>
   <ul>
      {% for a in authors %}
         <li>
            <a href="{{ a.id }}/">{{ a.name }}</a>
         </li>
      {% endfor %}
   </ul>
</body>
</html>
```

```
{{ text|escape|linkbreaks }}
{{ text|truncatewords:"30" }}
```

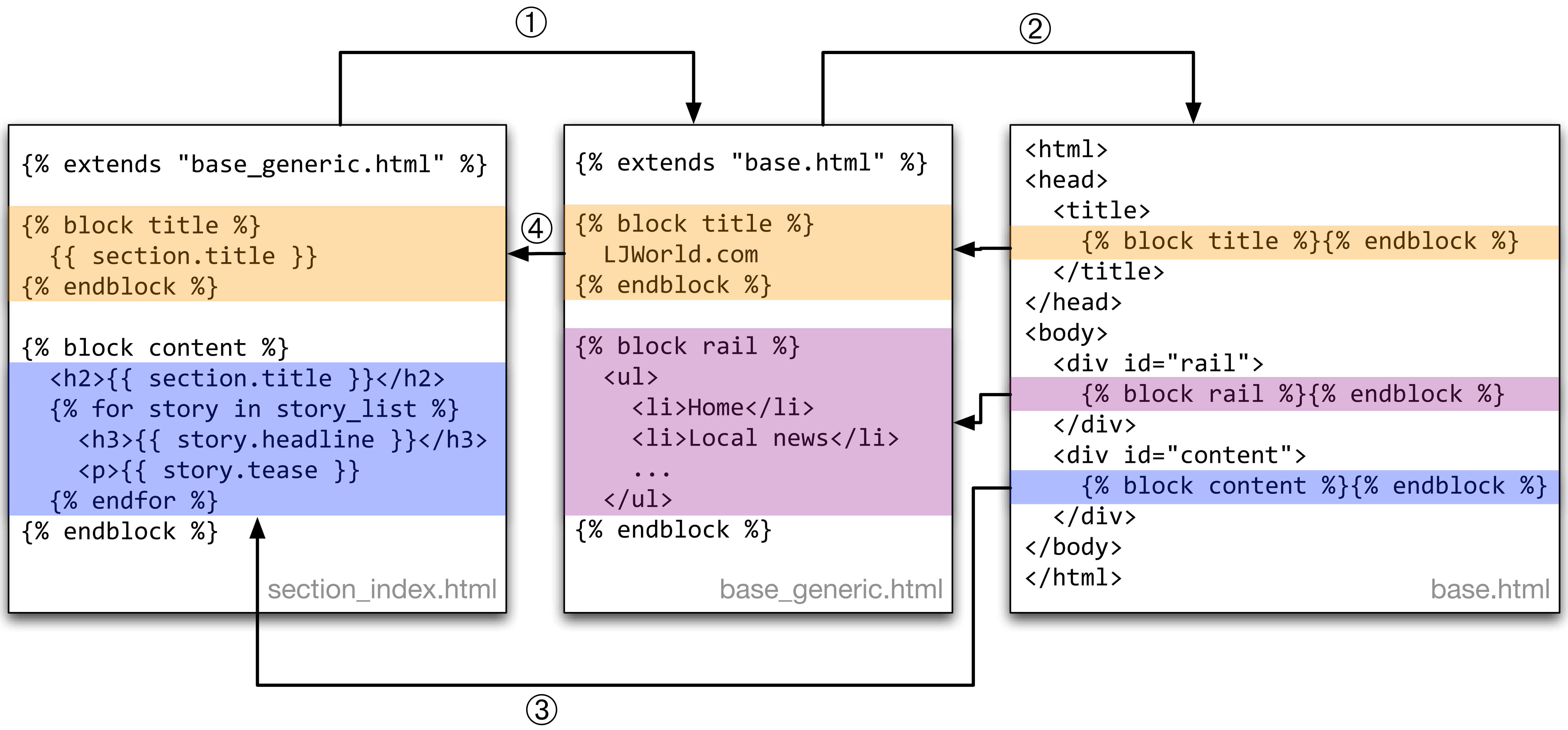
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Authors</title></head>
<body>
   <h1>Authors ({{ authors|length }} total)</h1>
   <ul>
      {% for a in authors %}
         <li>
            <a href="{{ a.id }}/">{{ a.name }}</a>
         </li>
      {% endfor %}
   </ul>
</body>
</html>
```

Template inheritance

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head>
   <title>
      {% block title %}YABL{% endblock %}
   </title>
</head>
<body>
   <div id="content">
      {% block content %}{% endblock %}
   </div>
   <div id="footer">
      {% block footer %}Copyright blah..{% endblock %}
   </div>
</body>
</html>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head>
   <title>
      {% block title %}YABL{% endblock %}
   </title>
</head>
<body>
   <div id="content">
      {% block content %}{% endblock %}
   </div>
   <div id="footer">
      {% block footer %}Copyright blah..{% endblock %}
   </div>
</body>
</html>
```

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

{% block title %}
   Authors | {{ block.super }}
{% endblock %}

{% block content %}
   <h1>Authors ({{ authors|length }} total)</h1>
   <ul>
      {% for a in authors %}
         <li>
            <a href="{{ a.id }}/">{{ a.name }}</a>
         </li>
      {% endfor %}
   </ul>
{% endblock %}
```



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

{% block title %}
  {{ section.title }}
{% endblock %}

{% block content %}
  <h2>{{ section.title }}</h2>
  {% for story in story_list %}
    <h3>{{ story.headline }}</h3>
    <p>{{ story.tease }}
  {% endfor %}
{% endblock %}
```

section_index.html

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

{% block title %}
  LJWorld.com
{% endblock %}

{% block rail %}
  <ul>
    <li>Home</li>
    <li>Local news</li>
    ...
  </ul>
{% endblock %}
```

base_generic.html

```
<html>
<head>
  <title>
    {% block title %}{% endblock %}
  </title>
</head>
<body>
  <div id="rail">
    {% block rail %}{% endblock %}
  </div>
  <div id="content">
    {% block content %}{% endblock %}
  </div>
</body>
</html>
```

base.html

Why?

Inheritance tips

- {% extends %} must be the first thing in your template.
- More {% block %}s are better.
- If you’re duplicating content, you’re missing a block.
- {{ block.super }}

Documentation

http://djangobook.com/en/2.0/chapter04/

http://django.me/templates

Exercise:

/authors/
/authors/{id}/

/entries/
/entries/{slug}/

REVSYS revolution systems

Django training

BONUS: Models and queries

Terminology

Models

a.k.a. “DDL”

Managers

a.k.a. “table”

QuerySets

a.k.a. “selection”

Model instances

a.k.a. “row”

```
>>> qs = Entry.objects.filter(is_published=True)
>>> qs
[<Entry: Man bites dog>, <Entry: Dog bites man>]

>>> e = qs[0]
>>> e
<Entry: Man bites dog>
```

Model	Manager	QuerySet	Instance

Models

Instance methods

```
class Entry(models.Model):
    …

    def is_by_jacob(self):
        return "jacob" in self.author.name.lower()

…

[1] >>> e = Entry.objects.get(pk=1)

[2] >>> e.is_by_jacob()
[2]   : False
```

“Special” instance methods

__unicode__

```
class Entry(models.Model):
    …

    def __unicode__(self):
        return self.headline

…

[1] >>> Entry.objects.all()
[1]   : [<Entry: Man bites dog>, <Entry: Dog bites man>]
```

save

```
class Entry(models.Model):
    …

    def save(self, **kwargs):
        self.word_count = count_words(self.body)
        super(Entry, self).save(**kwargs)
```

save

```
class Entry(models.Model):
    …

    def save(self, **kwargs):
        self.word_count = count_words(self.body)
        super(Entry, self).save(**kwargs)
```

Don't forget this part!

delete

```
class Author(models.Model):
    …

    def delete(self):
        nobody = Author.objects.get(first_name='<NOBODY>')
        self.entries.update(author=nobody)
        super(Author, self).delete()
```

Managers

Default manager

```
class Entry(models.Model):
    …

    objects = models.Manager()

…

[1] >>> from yabl.entries.models import Entry

[2] >>> Entry.objects
[2]   : <django.db.models.manager.Manager object at 0x7eca70>
```

Custom managers

```
class EntryManager(models.Manager):
    def future(self):
        …
    def past(self):
        …

class Entry(models.Model):
    …

    objects = EntryManager()
```

```
[1] >>> from yabl.entries.models import Entry

[2] >>> Entry.objects.future()
[2]   : [<Entry: Hi>]

[3] >>> Entry.objects.past()
[3]   : [<Entry: Man bites dog>, <Entry: Dog bites man>]
```

Documentation

http://django.me/managers

QuerySets

Filters

```
[1] >>> Author.objects.filter(first_name='Jacob')
[1]   : [<Author: Jacob Kaplan-Moss>]

[2] >>> Author.objects.filter(last_name__contains='s')
[2]   : [<Author: Miguel de Cervantes>, <Author: Jacob Kaplan-Moss>]

[3] >>> Author.objects.filter(last_name__contains='s', first_name='Miguel')
[3]   : [<Author: Miguel de Cervantes>]

[4] >>> Author.objects.filter(last_name__contains='s').filter(first_name='Miguel')
[4]   : [<Author: Miguel de Cervantes>]
```

Field lookups

`exact, iexact`	`name__exact='Joe'`
`contains, icontains`	`name__icontains='s'`

`startswith, endswith,`
`istartswith, iendswith`
	`name__endswith='nd'`

`in`	`name__in=('Joe', 'Jane')`
	`author__in=Author.objects.filter(…)`

`gt, gte, lt, lte`	`cost__gt=100`

`range`	`cost__range=(100, 500)`
	`date__range=(now, tomrrow)`

`year, month, day, week_day`	`date__year=2009`
	`date__month=7`

`isnull`	`author__isnull=True`
`regex, iregex`	`name__regex='^J.*b$'`

Following relationships

```
[1] >>> Entry.objects.filter(author__first_name__startswith='J')
[1]   : [<Entry: Hi>]

[2] >>> Author.objects.filter(entries__headline='Hi')
[2]   : [<Author: Jacob Kaplan-Moss>]
```

Where'd that come from?

related_name

```
class Entry(models.Model):
    author = models.ForeignKey(Author, related_name='entries')
```

select_related()

```
[1] >>> e = Entry.objects.get(pk=1)

[2] >>> e.author
[2]   : <Author: Jacob Kaplan-Moss>

Oops, that did a second, needless query.

[3] >>> e = Entry.objects.select_related().get(pk=1)

[4] >>> e.author
[5]   : <Author: Jacob Kaplan-Moss>

No second query needed for e.author
```

Limiting select_related()

```
[1] >>> Entry.objects.select_related('author', 'category')

[2] >>> Entry.objects.select_related(depth=2)
```

QuerySet details

QuerySets are chainable

```
[1] >>> Entry.objects.filter(
  ....:     headline__contains='bites',
  ....: ).exclude(
  ....:     pub_date__year=2008
  ....: ).filter(
  ....:     pub_date__month=9
  ....: )
```

QuerySets are unique

```
[1] >>> qs1 = Entry.objects.filter(headline__icontains='dog')

[2] >>> qs2 = qs1.exclude(pub_date__year=2008)

[3] >>> qs3 = qs1.filter(pub_date__year=2008)
```

QuerySets are lazy

```
[1] >>> qs = Entry.objects.filter(headline__icontains='dog')

[2] >>> qs = qs.exclude(pub_date__year=2008)

[3] >>> qs = qs.filter(author__first_name='Jacob')

[4] >>> qs
[4]   : [<Entry: Man bites dog>, <Entry: Dog bites man>]
```

When QuerySets are evaluated

Iteration	`for i in qs`
Slicing	`qs[0:5]`
Printing	`print qs, str(qs)`
len()	`len(qs)`
list()	`list(qs)`

Chainable methods

`filter(), exclude()`	`qs.filter(name='Joe')`
`order_by()`	`qs.order_by('-first_name')`
`reverse()`	`qs.reverse()`
`distinct()`	`qs.distinct()`
`values(), values_list()`	`qs.values('first_name', 'last_name')`

`dates()`	`qs.dates('pub_date', 'year')`
	`qs.dates('pub_date', 'month')`

`select_related()`	`qs.select_related()`

`defer(), only()`	`qs.defer('body')`
	`qs.only('body', 'headline')`

`none(), all()`	`qs.all()`
	`qs.none()`

Other QuerySet methods

`get()`	`e = Entry.objects.get(…)`
`create()`	`e = Entry.objects.create(…)`
`get_or_create()`	`e, created = Entry.objects.get_or_create(…)`
`count()`	`Entry.objects.count()`
`in_bulk()`	`Entry.objects.in_bulk([1, 2, 3])`
`latest()`	`Entry.objects.latest('pub_date')`

Raw SQL

```
[1] >>> query = "SELECT * FROM authors_author WHERE first_name = %s"

[2] >>> params = ["Jacob"]

[3] >>> Entry.objects.raw(query, params)
[3]     [<Person: Jacob Kaplan-Moss>]
```

```
Entry.objects.raw(query % params)
```

No!

Other topics

Aggregation

http://jacobian.org/r/django-aggregation

Transaction control

http://jacobian.org/r/django-transactions

Exercise:

/entries/future/
/entries/past/

What else?

- Forms, model forms, form sets, ...
- File storage - local and remote.
- Cookies, sessions, authn/authz.
- GeoDjango
- Built-in SQLi, XSS and CSRF protection.
- i18n and l10n support.
- Generic views,
- &c!

Thank you!

jacob**@jacobian**.org

http://lanyrd.com/sfypm