django

A (fast-paced) introduction

Jacob Kaplan-Moss

Colorado Software Summit
October 19th - 24th, 2008

What’s on the plate
What’s on the plate

→ A (quick) introduction
What’s on the plate

➡️ A (quick) introduction
➡️ Creating models
What’s on the plate

➡️ A (quick) introduction
➡️ Creating models
➡️ The automatic admin interface
What’s on the plate

- A (quick) introduction
- Creating models
- The automatic admin interface
- Views (and URLs)
What’s on the plate

- A (quick) introduction
- Creating models
- The automatic admin interface
- Views (and URLs)
- Templates
Introduction
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.”
More...

http://docs.djangoproject.com/en/dev/misc/design-philosophies/
http://www.djangobook.com/en/1.0/chapter01/
Installation

http://docs.djangoproject.com/en/dev/intro/install/
http://www.djangobook.com/en/1.0/chapter02/
http://code.djangoproject.com/wiki/SetupOnTiger
http://code.djangoproject.com/wiki/WindowsInstall
“Projects”
Our project:
# Docutils

**Docutils** -- Python Documentation Utilities

Docutils is a modular system for processing documentation into useful formats, such as HTML, XML, and LaTeX. For input, Docutils supports ReStructuredText, an easy-to-read, what-you-see-is-what-you-get plaintext markup syntax.

![Moldy and Fresh buttons]

| +5 |

## Author

David Goodger

## Links

- Home page
- Cheeseshop page

## Categories

- Development Status :: 3 - Alpha
- Environment :: Console
- Intended Audience :: Developers
- Intended Audience :: End Users/Desktop
- Intended Audience :: Other Audience
- Intended Audience :: System Administrators
- License :: OSI Approved :: BSD License
- License :: OSI Approved :: GNU General Public License (GPL)
- License :: OSI Approved :: Python Software Foundation License
- License :: Public Domain
- Natural Language :: English
- Natural Language :: Esperanto
- Natural Language :: French
- Natural Language :: German
- Natural Language :: Italian
- Natural Language :: Russian
- Natural Language :: Slovak
- Natural Language :: Spanish
- Natural Language :: Swedish
- Operating System :: OS Independent
- Programming Language :: Python
- Topic :: Documentation
- Topic :: Software Development :: Documentation
- Topic :: Text Processing

http://www.cheeserater.com/
http://www.cheeserater.com/

http://github.com/jacobian/cheeserater/tree
$ django-admin.py startproject cheeserater
cheeserater/
  __init__.py
manage.py
settings.py
urls.py
$ ./manage.py runserver
Validating models...
0 errors found.

Django version 1.0, using settings 'cheeserater.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 cheeserater/settings.py.
- Start your first app by running python cheeserater/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
$ ./manage.py syncdb
Creating table auth_message
Creating table auth_group

... 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.

... Adding permission 'user | Can add user'
Adding permission 'user | Can change user'
Adding permission 'user | Can delete user'

...
More...

http://docs.djangoproject.com/en/dev/topics/settings/
http://docs.djangoproject.com/en/dev/ref/django-admin/
“Apps”
Models
What’s a model?
CREATE TABLE "packages_package" (  
  "id" integer NOT NULL PRIMARY KEY,  
  "name" varchar(300) NOT NULL,  
  "version" varchar(300) NOT NULL,  
  "home_page" varchar(200) NOT NULL,  
  "summary" text NOT NULL,  
  "description" text NOT NULL,  
  "keywords" text NOT NULL  
);

Scary Quirky Language
Scary Quirky Language

→ SQL is tough
Scary Quirky Language

- SQL is tough
- SQL knows no version control
Scary Quirky Language

- SQL is tough
- SQL knows no version control
- DRY
Scary Quirky Language

- SQL is tough
- SQL knows no version control
- DRY
- Python is fun!
Defining Models
$ ./manage.py startapp packages
INSTALLED_APPS = (
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.sites",
    "cheeserater.packages"
)
from django.db import models

class Package(models.Model):
    name = models.CharField(max_length=300)
    version = models.CharField(max_length=300, blank=True)
    author = models.CharField(max_length=300, blank=True)
    home_page = models.URLField(blank=True)
    summary = models.TextField()
    description = models.TextField(blank=True)
    keywords = models.TextField(blank=True)
    categories = models.ManyToManyField(Category, related_name="packages")
$ ./manage.py validate
0 errors found.
$ ./manage.py syncdb
Creating table packages_package
...
Model API
$ ./manage.py shell

```python
>>> from cheeserater.packages.models import Package
>>> p = Package(
...    name="Django",
...    version="1.0")
>>> p.save()
```
>>> ps = Package.objects.all()

>>> ps
[
<Package: Package object>,
]

>>> ps[0].name
'u'Django'

>>> p = Package.objects.get(name__contains="Dj")

>>> p.version
'u'1.0'
Model metadata
class Package(models.Model):
    ...

    def __unicode__(self):
        return self.name
class Package(models.Model):
    ...

class Meta:
    ordering = ["name"]
    verbose_name = "cheeseshop package"
    verbose_name_plural = "cheeseshop packages"
Relationships
class Topic(models.Model):
    name = models.CharField(max_length=150, unique=True)
    slug = models.SlugField()

class Category(models.Model):
    topic = models.ForeignKey(Package,
                               related_name="categories")
    value = models.CharField(max_length=100)
    slug = models.SlugField()}
>>> license = Topic.objects.get(name="License")
>>> license.categories.all()
[<Category: License :: DFSG approved>, ...]
>>> license.categories.filter(value__startswith="OSI").count()
17L

>>> bsd = license.categories.get(value__contains="BSD")
>>> bsd.topic
<Topic: License>
>>> bsd.packages.count()
210L

>>> p = Package.objects.get(name="BeautifulSoup")
>>> p.categories.all()
[<Category: Programming Language :: Python>, ...]
>>> p.categories.get(topic=license)
<Category: License :: OSI Approved :: Python Software Foundation License>

>>> p.categories = [bsd]
>>> p.categories.all()
[<Category: License :: OSI Approved :: BSD License>]
Show all Spanish-language packages with a BSD license.
>>> packages = Package.objects.filter(
... categories__topic__name = "Natural Language",
... categories__value__contains = "Spanish",
... categories__topic__name = "License",
... categories__value__contains = "BSD"
... )

>>> packages
[<Package: TaskCoach>, ...]
```python
>>> packages = Package.objects.filter(
...     categories__topic__name = "Natural Language",
...     categories__value__contains = "Spanish",
... )

>>> packages = packages.filter(
...     categories__topic__name = "License",
...     categories__value__contains = "BSD",
... )

>>> packages
[<Package: TaskCoach>, ...]
```
More...

http://docs.djangoproject.com/en/dev/topics/db/models/
http://docs.djangoproject.com/en/dev/topics/db/queries/
Django Admin
What is it?
Activate!
from django.contrib import admin
from cheeserater.packages.models import Package

admin.site.register(Package)
from django.contrib import admin
from cheeserater.packages.models import Package

class PackageAdmin(admin.ModelAdmin):
    list_display = ('name', 'version', 'summary')
    search_fields = ('name', 'summary', 'keywords')

admin.site.register(Package, PackageAdmin)
INSTALLED_APPS = ( 
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.sites",
    "django.contrib.admin",
    "cheeserater.packages",
)
from django.conf.urls.defaults import *
from django.contrib import admin

admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/(.*)', admin.site.root,
    )}
$ ./manage.py runserver
More...

http://docs.djangoproject.com/en/dev/intro/tutorial02/
http://docs.djangoproject.com/en/dev/ref/contrib/admin/
Views
What’s a view?
URLs
page.php

script.cgi?pageid=144

StoryPage.aspx
/packages/
/packages/Unipath/
/packages/reportlab/
ROOT_URLCONF = "cheeserater.urls"
from django.conf.urls.defaults import *
from django.contrib import admin
from cheezerater.packages.views import *

urlpatterns = patterns('',
    (r'^admin/(.*)', admin.site.root),
    ('^packages/$', package_list),
    ('^packages/(?P<package_name>.*)/$', package_detail),
)

admin.autodiscover()
Dissecting a request

- GET /packages/Unipath/
- ROOT_URLCONF
- cheeserater.urls
- ('^packages/([^<package_name>].*)/$', package_detail)
- package_detail(request, package_name='Unipath')
More...

http://docs.djangoproject.com/en/dev/topics/http/urls/
A first view
from django.http import HttpResponse

def package_list(request):
    return HttpResponse("This is the package list!")
from django.http import HttpResponse
from cheeserater.packages.models import Package

def package_list(request):
    r = "<ul>"
    ps = Package.objects.order_by("name")
    for p in ps:
        r += "<li>%s: %s</li>" % \
            (p.name, p.description)
    r += "</ul>"
    return HttpResponse(r)
from django import template
from django.http import HttpResponse
from cheeserater.packages.models import Package

def package_list(request):
    ps = Package.objects.order_by("name")
    t = template.loader.get_template("packages/index.html")
    c = Context({"package_list": ps})
    return HttpResponse(t.render(c))
from django.shortcuts import render_to_response
from cheeseater.packages.models import Package

def package_index(request):
    ps = Package.objects.order_by("name")
    return render_to_response("packages/index.html",
                              {"package_list" : ps})
Templates
What’s a template?
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Packages</title></head>
<body>
<h1>Packages ({{ package_list|length }} total)</h1>
<ul>
{% for p in package_list %}
<li>
  <a href="{{ p.name|urlencode }}/">
    {{ p.name }}
  </a>
</li>
{% endfor %}
</ul>
</body>
</html>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Packages</title></head>
<body>
<h1>Packages ({{ package_list|length }} total)</h1>
<ul>
{% for p in package_list %}
<li>
  <a href="{{ p.name|urlencode }}/">
    {{ p.name }}
  </a>
</li>
{% endfor %}
</ul>
</body>
</html>
The magic dot

- `p["name"]`
- `p.name`
- `p.name()`
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Packages</title></head>
<body>
<h1>Packages ({{ package_list|length }} total)</h1>
<ul>
{% for p in package_list %}
<li>
<a href="{{ p.name|urlencode }}/">
{{ p.name }}
</a>
</li>
{% endfor %}
</ul>
</body>
</html>
{{ text|truncatemaxwords:"30" }}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head><title>Packages</title></head>
<body>
<h1>Packages ({{ package_list|length }} total)</h1>
<ul>
{% for p in package_list %}
<li>
<a href="{{ p.name|urlencode }}/">{{ p.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 %}CheeseRater{% 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 %}CheeseRater{% 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 %}
   Packages | {{ block.super }}
{% endblock %}

{% block content %}
   <h1>Packages ({{ package_list|length }} total)</h1>
   <ul>
      {% for p in package_list %}
      <li>
         <a href="{{ p.name|urlencode }}/">
            {{ p.name }}
         </a>
      </li>
      {% endfor %}
   </ul>
{% endblock %}
Why?
Inheritance tips
Inheritance tips

→ `{% extends %}` must be the first thing in your template.
Inheritance tips

- `{% extends %}` must be the first thing in your template.

- More `{% block %}`s are better.
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.
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 }}`
More...

http://docs.djangoproject.com/en/dev/topics/templates/
What else ya’ got?
What else ya’ got?

➡️ Generic views
What else ya’ got?

- Generic views
- Testing tools
What else ya’ got?

→ Generic views
→ Testing tools
→ Forms
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
- İnternâtîonâlizætiøn
- RSS
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
- İnternationálization
- RSS
- Sitemaps
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
- Internationálizætiøn
- RSS
- Sitemaps
- Support for legacy databases
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
- Ânternâtiônàlizætiøn
- RSS
- Sitemaps
- Support for legacy databases
- Automatic CSRF protection
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching
- Êntërnåtiônàlizætiøn
- RSS
- Sitemaps
- Support for legacy databases
- Automatic CSRF protection
- GIS
What else ya’ got?

- Generic views
- Testing tools
- Forms
- Custom template tags and filters
- Sessions
- Authentication/authorization
- Caching

- Internätiônàlizætiön
- RSS
- Sitemaps
- Support for legacy databases
- Automatic CSRF protection
- GIS
- Pluggable file storage
Get involved!

http://www.djangoproject.com/

http://groups.google.com/group/django-users

jacob@jacobian.org