geodjango

Rapid Geographic Web Application with GeoDjango

May 14, 2008
Where 2.0
Django Intro
Introduction
Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.
More…

http://toys.jacobian.org/presentations/2008/pycon/tutorial/
http://www.djangoproject.com/documentation/
design_philosophies/
http://www.djangobook.com/en/1.0/chapter01/
$ django-admin.py startproject where2
where2/
  __init__.py
  manage.py
  settings.py
  urls.py
from django.db import models

class County(models.Model):
    name = models.CharField(...)

>>> County(name='x').save()
>>> County(name='y').save()
>>> County.objects.count()
2
>>> c = County.objects.get(name='y')
CREATE TABLE "app_county" (  
    "id" integer NOT NULL PRIMARY KEY,  
    "name" varchar(50) NOT NULL  
);
from django.contrib.gis.db import models
from django.contrib.gis.geos import Point

class County(models.Model):
    name = models.CharField(...)
    center = models.PointField(srid=4269)
    objects = models.GeoManager()

>>> p1 = 'POINT (0 1)'
>>> p2 = Point(10, 20)
>>> County(name='x', center=p1).save()
>>> County(name='y', center=p2).save()
>>> County.objects.count()
2
>>> y = County.objects.get(center=p2)
CREATE TABLE "app_county" (  
    "id" integer NOT NULL PRIMARY KEY,  
    "name" varchar(50) NOT NULL NULL,  
);  
SELECT AddGeometryColumn( 'app_county',  
    'center', 4269,  
'POINT', 2  
);
$ ./manage.py syncdb
Creating table auth_message
Creating table auth_group
...
Creating table app_county
...

$. ./manage.py runserver 0:8000
Validating models...
0 errors found.

Django version 0.97-pre-SVN-7400, using settings 'settings'
Development server is running at http://0:8000/
Quit the server with CONTROL-C.
“Apps”

http://www.b-list.org/weblog/2008/mar/15/slides/
“Views”

The guts.
Dissecting a request

- GET /some_url/
- settings.ROOT_URLCONF = 'urls'
- urls
- (r'^some_url/$', include('app.urls'))
- app.urls
- ('^$', county_list)
- county_list(request)
from django.conf.urls.defaults import *
from app.views import *

urlpatterns = patterns('',
    ('^app/$', county_list),
)
from django.shortcuts import render_to_response
from models import County

def county_list(request):
    cs = County.objects.order_by('name')
    return render_to_response('county_list.html',
    {‘counties’:cs})
Templates
Skinning
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
<head>
<title>Counties</title>
</head>
<body>
  <ul>
    {% for c in counties %}
    <li>{{ c.name }}</li>
    {% endfor %}
  </ul>
</body>
</html>
The magic dot

- `p["name"]`
- `p.name`
- `p.name()`
More...

http://www.djangoproject.com/documentation/
http://www.djangoproject.com/documentation/
http://www.djangoproject.com/documentation/

No, really.

http://www.djangoproject.com/documentation/
Installation: it builds character

- Relying on relatively new versions of the most libraries.
- But ctypes saves us (sort-of).
- We plan to fix this at some point.
- For now, hurrah for virtualization!
Third-Party Libraries
Third-Party Libraries

- GEOS
- GDAL
- GeoIP (BSD-licensed)
Third-Party Libraries

• Why?
  • Powerful open source libraries; temperamental SWIG interfaces
  • ctypes enables all-Python interfaces (no compilation necessary)
  • Use of C APIs allows for high degree of cross-platform compatibility
>>> from django.contrib.gis.geos import *
>>> pnt = Point(5, 23)
>>> ring = LinearRing((0, 0), (0, 50), (50, 50), (50, 0), (0, 0))
>>> poly = Polygon(ring)
>>> print poly.contains(pnt)
True
>>> print poly
POLYGON ((0.0000000000000000 0.0000000000000000, 0.0000000000000000 50.0000000000000000, 50.0000000000000000 50.0000000000000000, 50.0000000000000000 0.0000000000000000, 0.0000000000000000))
>>> print poly.kml
<Polygon><outerBoundaryIs><LinearRing><coordinates>0.0,0.0,0 0.0,50.0,0 50.0,50.0,0 50.0,0.0,0 0.0,0.0,0</coordinates></LinearRing></outerBoundaryIs></Polygon>
>>> from django.contrib.gis.gdal import *
>>> s1 = SpatialReference(4326)
>>> s2 = SpatialReference('NAD83')
>>> s3 = SpatialReference('+proj=lcc +lat_1=27.5 +lat_2=35
  +lat_0=18 +lon_0=-100 +x_0=1500000 +y_0=5000000 +ellps=GRS80
  +units=m +no_defs')
>>> geom = OGRGeometry('POINT(5 23)', s1)
>>> geom.transform(900913)
>>> print geom
POINT (556597.453966367174871 2632018.637504272162914)
>>> from django.contrib.gis.utils import GeoIP
>>> g = GeoIP()
>>> print g.country('refractions.net')
{'country_name': 'Canada', 'country_code': 'CA'}
>>> print g.city('refractions.net')
{'city': 'Vancouver', 'region': 'BC', 'area_code': 0,
 'longitude': -123.13330078125, 'country_code3': 'CAN',
 'latitude': 49.25, 'postal_code': 'v6c2b5', 'dma_code': 0,
 'country_code': 'CA', 'country_name': 'Canada'}
>>> print g.geos('refractions.net')
POINT (-123.1333007812500000 49.2500000000000000)

*This is not enabled on the VM.*
Inspection & Import
Gameplan

• Delete all states (so we have room to play)
• Inspect shapefile
  • ogrinfo - human-readable
  • ogrinspect - model creation
• Define model
• Load with LayerMapping
Delete states

```python
>>> from census.models import State
>>> State.objects.all().delete()
```
>>> from django.contrib.gis.utils \
... import ogrinfo, ogrinspect
>>> ogrinfo(df('st'), num_features=2)
...
>>> print ogrinspect(df('st'), \
...    'State', srid=4269)
...
Define Model

class State(models.Model):
    #STATE
    fips = models.CharField(max_length=2)
    #NAME
    name = models.CharField(max_length=20)
    #MULTIPOLYGON
    mpoly = models.MultiPolygonField(srid=4269)
    objects = models.GeoManager()
Define

```python
>>> mapping = {
...    'fips': 'STATE',
...    'name': 'NAME',
...    'mpoly': 'MULTIPOLYGON'
... }

>>> lm = LayerMapping(State, \
...    df('st'), mapping, \
...    unique=('name', 'fips'), \
...    encoding='cp437', \
...    transform=False)

>>> lm.save(verbosе=True)
```
Your turn:
Load Counties!
Ask questions.
Exploration
Spatial Queries

Neighborhood.objects.filter(poly__intersects=zipcode.mpoly) | Neighborhood.objects.filter(poly__within=county.mpoly)

VS.

SELECT "houston_neighborhood"."id", "houston_neighborhood"."name", "houston_neighborhood"."poly"
FROM "houston_neighborhood" WHERE (ST_Intersects("houston_neighborhood"."poly", ST_Transform(ST_GeomFromWKB('\001\006\000 ... ', 4269), 32140)) OR ST_Within("houston_neighborhood"."poly", ST_GeomFromWKB('\001\006\000 ... ', 32140)))
Spatial Queries

• Available PostGIS lookup types:

  • overlaps, bboverlaps
  • overlaps_left, overlaps_right
  • overlaps_below, overlaps_above
  • strictly_below, strictly_above
  • left, right
  • same_as/exact
  • contained, bbcontains
  • equals, disjoint, touches, crosses, within, intersects, relate
Distance Queries

• Projected/Geodetic coordinate system a source of confusion for beginners.

• Inherent PostGIS limitation,

• Distance Lookups:
  • distance_lte, distance_lt
  • distance_gte, distance_gt
Distance Queries

# Distances will be calculated from this point, # which does _not_ have to be projected.
>>> pnt = fromstr('POINT(-96.876369 29.905320)', srid=4326)

# If numeric parameter, units of field (meters in this case) # are assumed.
>>> qs = SouthTexasCity.objects.filter(
    point__distance_lte=(pnt, 7000))

# Find all Cities w/in 7km of pnt
>>> qs = SouthTexasCity.objects.filter(
    point__distance_lte=(pnt, D(km=7)))

# Find all Cities > 20 miles away from pnt.
>>> qs = SouthTexasCity.objects.filter(
    point__distance_gte=(pnt, D(mi=20)))

# More obscure units, such as chains, are supported.
>>> qs = SouthTexasCity.objects.filter(
    point__distance_gte=(pnt, D(chain=100)))
Distance Queries

• **Distance** object eases conversion between units of measure.

```python
>>> from django.contrib.gis.measure import Distance
>>> dist = Distance(ft=5280)
>>> print dist.mi
1.0
```

• Projected/Geodetic coordinate system a source of confusion for beginners.
Automatic Transformation

- If a geometry with a different SRID is used, it will be automatically transformed -- one less thing to worry about.

```python
>>> pnt = Point(-95.4067, 29.7183, srid=4326)
>>> qs = Neighborhood.objects.filter(mpoly__intersects=pnt)
>>> print qs.query.as_sql()
SELECT "texas_neighborhood"."id",
"texas_neighborhood"."name", "texas_neighborhood"."state",
"texas_neighborhood"."city", "texas_neighborhood"."county",
"texas_neighborhood"."region", "texas_neighborhood"."mpoly"
FROM "texas_neighborhood" WHERE
ST_Intersects("texas_neighborhood"."mpoly", ST_Transform(%s, 3084))
```
GeoQuerySet Methods

- gml
  - County.objects.all().gml
  - Neighborhood.objects.all().gml()
  - Neighborhood.objects.all().gml[0].gml
  - <gml:MultiPolygon srsName="EPSG:3084">...
    </gml:MultiPolygon>
GeoQuerySet Methods

- `kml`
- `Neighborhood.objects.all().kml()[0].kml`
- `<MultiGeometry><Polygon><outerBoundaryIs>...</MultiGeometry>"
GeoQuerySet Methods

- distance
  - `p=Neighborhood.objects.all()[2].mpoly.centroid`
  - `Neighborhood.objects.all().distance(p)[0].distance`
“Old” Admin

How you currently create an admin interface in trunk:

```python
from django.contrib.gis.db import models

class Location(models.Model):
    name = models.CharField(max_length=30)
    point = models.PointField()

class Admin:
    list_display = ['name']
    search_fields = ['name']
```
“Old” Admin

Old style admin URLs:

urlpatterns = patterns('',
    (r'^admin/', include('django.contrib.admin.urls')),
)
"Old" Admin

Change location

Name: FooDawg

Point: POINT (5.0000000000000000
23.0000000000000000)
newforms-admin

- The newforms-admin branch decouples Admin settings from your models.
- More flexibility and customization
- Other goal is to convert the Admin to use Django’s “newforms.”
• Django has branch policy prohibiting merges between SVN branches.

• I couldn’t wait for new functionality -- so I created mercurial merge between two branches.
How to create new admin interface:

```python
from django.contrib.gis import admin
from django.contrib.gis.db import models

class Location(models.Model):
    name = models.CharField(max_length=30)
    point = models.PointField()

class LocationAdmin(admin.GeoModelAdmin):
    list_display = ['name']
    search_fields = ['name']
```
How to create new admin interface (in urls.py):

```python
from django.contrib.gis import admin
from geoapp.models import Location
from geoapp.admin import LocationAdmin

admin.site.register(Location, LocationAdmin)

urlpatterns = patterns('',
    (r'^admin/(.*)', admin.site.root),
)
gis-newforms

Included in your VM as the default distribution.

May change to GIS SVN trunk via:
$ sudo chdjango.py gis

To change back:
$ sudo chdjango.py gis-newforms
Mapping
We will leave the presentation to explore a mini app (included on your VM) that shows TABC (Texas Alcoholic Beverage Commission) license permits in a particular Houston neighborhood.

Neighborhood data provided by Zillow ®.
Conclusion