Introduction to GeoDjango

David Wilson

An Outline:

- Who am I?
- What is GeoDjango?
- A Tour of GeoDjango
- Example: Colorado Geology

Who Am I?

Who Am I?

- Full time Java developer
- Love Python and the web
- Degree in Geophysics
- Some GIS background, actively working to improve knowledge
- Travel, weight lifting, cycling, and skiing

What is GeoDjango?

GeoDjango = spatial toolbox

Spatial?

Not Spatial

```
Customer.objects.filter(zipcode=zipcode)

Listing.objects.filter(city=city)

Shrubbery.objects.filter(location=robert_the_shrubber)
```

Spatial!

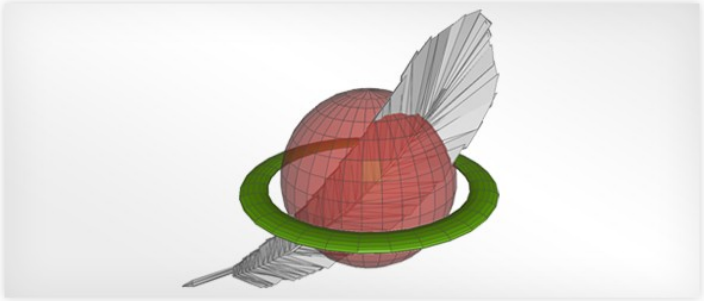
```
Customer.objects.filter(location__within=city)

Listing.objects.filter(poly__contains=pnt)

robert = Shrubber.objects.get(name="Robert")
Shrubbery.objects.filter(
    location__distance_lte=(robert.location, 500))
```

A Tour of GeoDjango

Tour: Backends

MySQL

PostGIS
Spatial PostgreSQL

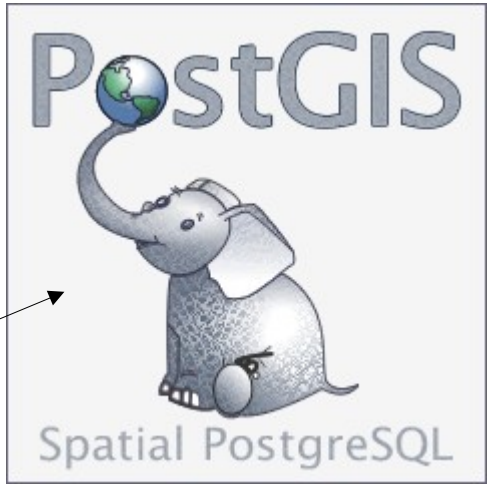


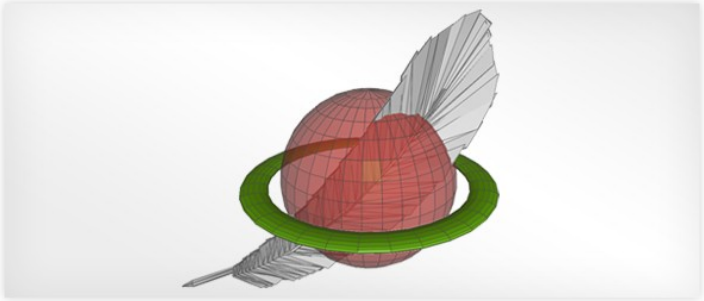
ORACLE

Tour: Backends

MySQL

Use this one!





ORACLE®

Tour: Models

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

Tour: Models

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

unit_srid = 4326

class Unit(models.Model):
    name = models.CharField(max_length=1024)
    description = models.CharField(max_length=1024)
    geom = models.GeometryField(srid=unit_srid)
    objects = models.GeoManager()

    def __unicode__(self):
        return "Unit %s" % (name)
```

Tour: Model Fields

- GeometryField
- PointField
- LineStringField
- PolygonField
- MultiPointField
- MultiLineStringField
- MultiPolygonField
- GeometryCollectionField

Tour: GeoManager

- Overrides model manager
- Enables spatial queries
- Gotcha: required on models with no spatial fields that need to query related models spatially

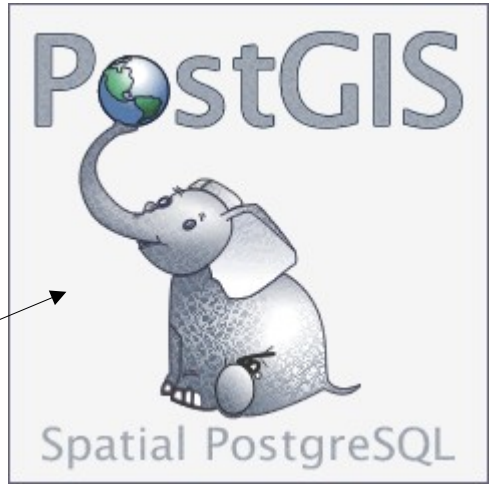
Tour: GeoQuerySet

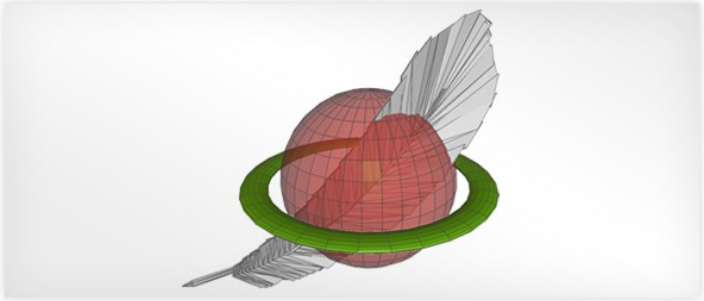
```
Unit.objects.filter(geom__intersects=geom)
```

- Many spatial queries, e.g.
  - contains
  - crosses
  - overlaps
  - intersects
  - Distance and area queries
- Gotcha: query availability varies based on backend

Tour: GeoQuerySet

Use this one!





ORACLE®

Tour: GEOS API

- Geometry Engine – Open Source
- Loosely coupled to GeoDjango
- Allows for simple creation, manipulation, and transformation of geometries

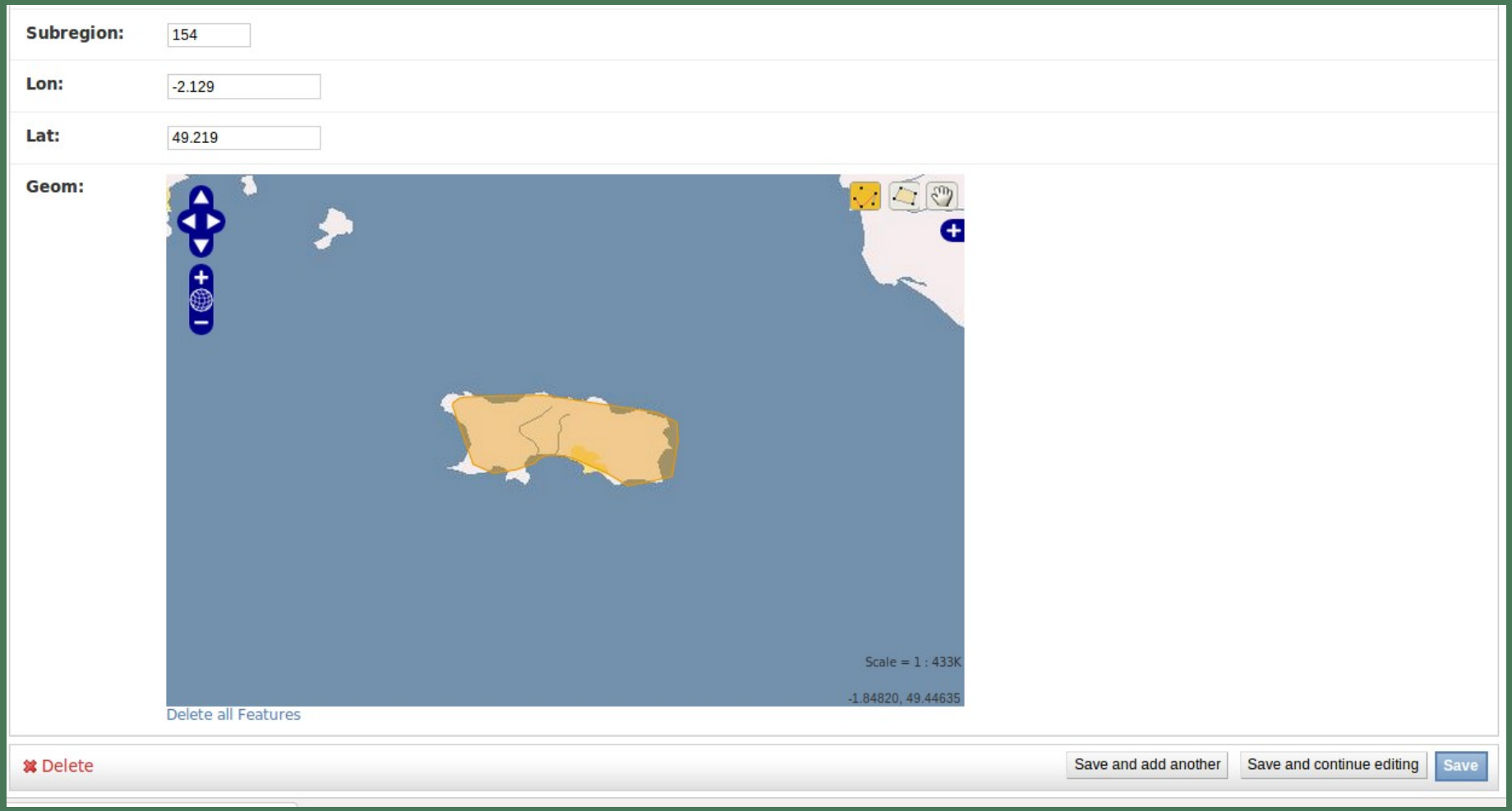
Tour: GDAL API

- Geospatial Data Abstraction Library
- Reads spatial data:
  - Shapefiles
  - KML
  - And many, many more!
- Easy to abstract data sources

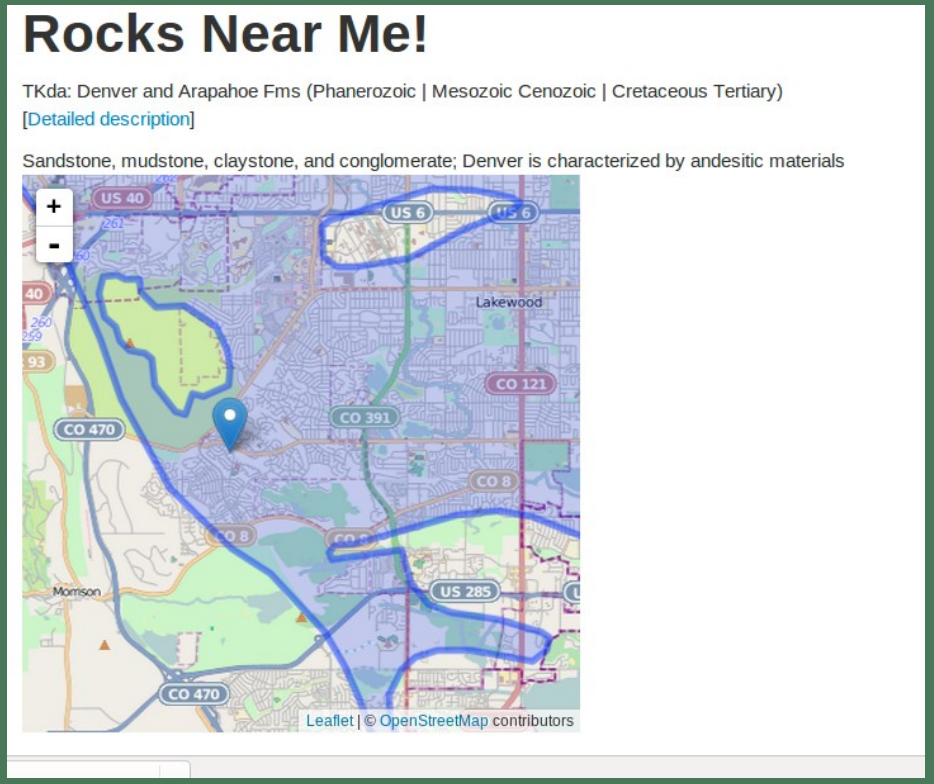
Tour: GeoIP

- Ctypes wrapper for MaxMind GeoIP
- Requires appropriate local dataset
- Allows server-side location
- Generally not as accurate as HTML5 location API

Tour: GeoAdmin



Example: Colorado Geology



Example: Objectives

- Find Colorado geology data
- Import data
- Query data (find rocks near you!)
- Make a pretty map

Example: Finding Data

Mineral Resources > Online Spatial Data > Geology > by state

Colorado geologic map data

A GIS database of geologic units and structural features in Colorado, with lithology, age, data structure, and format written and arranged just like the other states.

[Legend for lithology colors]

View:

Show in a web browser window:
http://mrdata.usgs.gov/sgmc/co.html

Show in Google Earth or download KML:

cogeol.kml (13.9M bytes)
cogeol.kmz (4.6M bytes)

Example: Importing Data

```
$ python manage.py ogrinspect cogeol.kml Unit --srid=4326 \
    --mapping --multi

from django.contrib.gis.db import models

class Unit(models.Model):
    name = models.CharField(max_length=1024)
    description = models.CharField(max_length=1024)
    geom = models.GeometryField(srid=4326)
    objects = models.GeoManager()

# Auto-generated `LayerMapping` dictionary for Unit model
unit_mapping = {
    'name' : 'Name',
    'description' : 'Description',
    'geom' : 'UNKNOWN',
}
```

Example: Importing Data

```
import os
import units

from django.core.management.base import BaseCommand, CommandError
from django.contrib.gis.utils import LayerMapping

from units.models import Unit, unit_mapping

class Command(BaseCommand):
    help = 'Loads geologic unit data from app data directory'

    def handle(self, *args, **options):
        unit_shp = os.path.abspath(
            os.path.join(os.path.join(os.path.dirname(units.__file__),
            'data/cogeol.kml')))

        lm = LayerMapping(Unit, unit_shp, unit_mapping,
            transform=False, encoding='iso-8859-1')
        lm.save(strict=True, verbose=True)
```

Example: View

```
def find_rocks(request):
    """
    Given a given lat/long pair, return the unit(s) surrounding it.
    """
    if request.is_ajax():
        lat = request.GET.get('lat', None)
        lon = request.GET.get('lon', None)

        if lat and lon:
            point = Point(float(lon), float(lat))
            units = Unit.objects.filter(geom__contains=point)
            geojson_data = GeoJSONSerializer().serialize(
                units, use_natural_keys=True)

            return HttpResponse(geojson_data,
                mimetype='application/json')
    msg = "Bad request: not AJAX or no latlong pair present"
    return HttpResponseBadRequest(msg)
```

Example: JavaScript

```
var getLocation = function() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(handlePosition);
        } else {
            loc.html("GeoLocation not supported");
        }
    }
var handlePosition = function(position) {
    var url = "{% url 'find-rocks' %}";
    var lat = position.coords.latitude;
    var lon = position.coords.longitude;
    $.get(url, {"lat": lat,
                "lon": lon },
                function(data) {
                    rocks.empty();
                    $.each(data.features, function(index, val) {
                        var name = val.properties.name;
                        var description = val.properties.description;
                        rocks.append(name);
                        rocks.append(description);
                        makeMap(val, lon, lat);
                    });
                });
    }
```

Example: More JavaScript

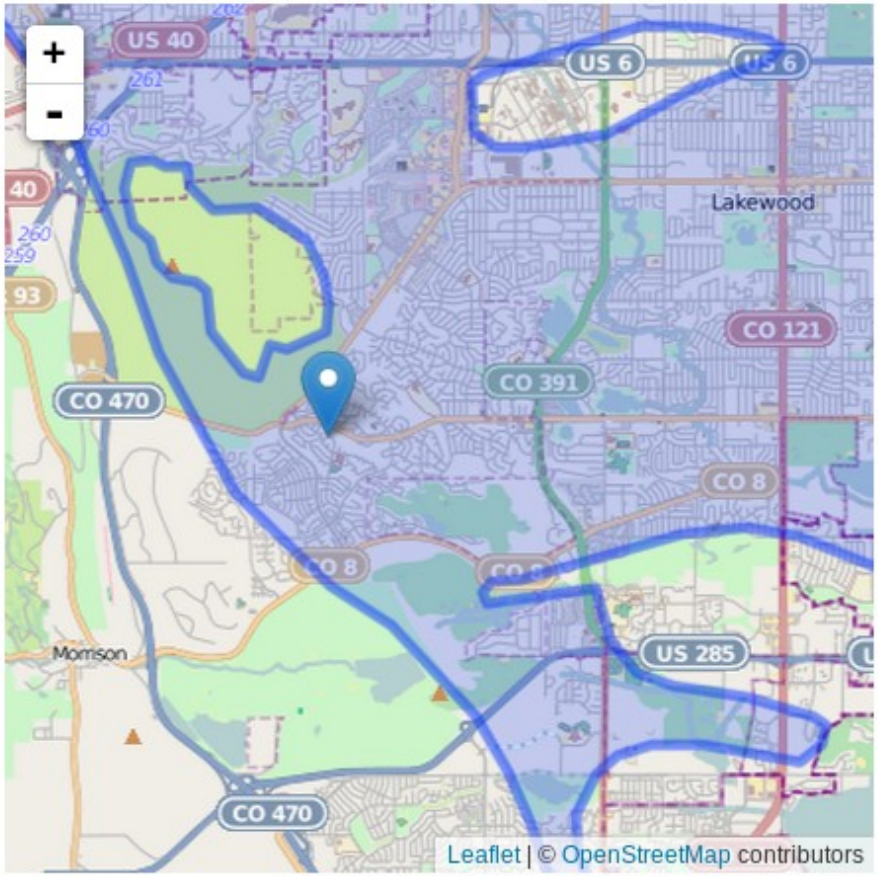
```
var featurePopup = function(feature, layer) {
        var popupContent = feature.properties.name;

        if (feature.properties && feature.properties.popupContent) {
            popupContent += feature.properties.popupContent;
        }

        layer.bindPopup(popupContent);
    }



var makeMap = function(feature, lon, lat) {
    var map = L.map('map').setView([lat, lon], 13);

    //Unit
    var unit = L.geoJson(feature, {
        onEachFeature: featurePopup
    }).addTo(map);

    //OSM layer
    L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
    }).addTo(map);

    // Marker
    L.marker([lat, lon]).addTo(map)
        .bindPopup('You are here.')
        .openPopup();
}
```

Example: Demo

Rocks Near Me!

TKda: Denver and Arapahoe Fms (Phanerozoic | Mesozoic Cenozoic | Cretaceous Tertiary)
[Detailed description]

Sandstone, mudstone, claystone, and conglomerate; Denver is characterized by andesitic materials



Conclusions

Conclusions

- Spatial toolbox
- Use the parts you need
- Built into the same Django you already own!
- Will need to understand GIS basics
- I hope to clean up this demo app and make a full tutorial

Resources

Docs:
https://docs.djangoproject.com/en/dev/ref/contrib/gis/

Vagrant image:
https://github.com/david-wilson/vagrant-geodjango-base

USGS Colorado Geology Data:
http://mrdata.usgs.gov/geology/state/state.php?state=CO

LeafletJS:
http://leafletjs.com/

Contact Me:
david@davidwilson.me, @dwilson86, http://davidwilson.me