Maps of Imaginary Lands

Malcolm Tredinnick

GeoDjango Admin

Nice experience out of the box

GeoDjango admin?!

How does it work?

What can I do next?

Is this all there is?

OpenLayers

✓ Client side, Javascript framework

✓ Combines data from multiple data feeds

✓ Provides neat looking UI around it

✓ Day to learn, lifetime to customise

Mapnik

- ✓ Server side way to combine data sources

- ✓ Different details and different zoom levels
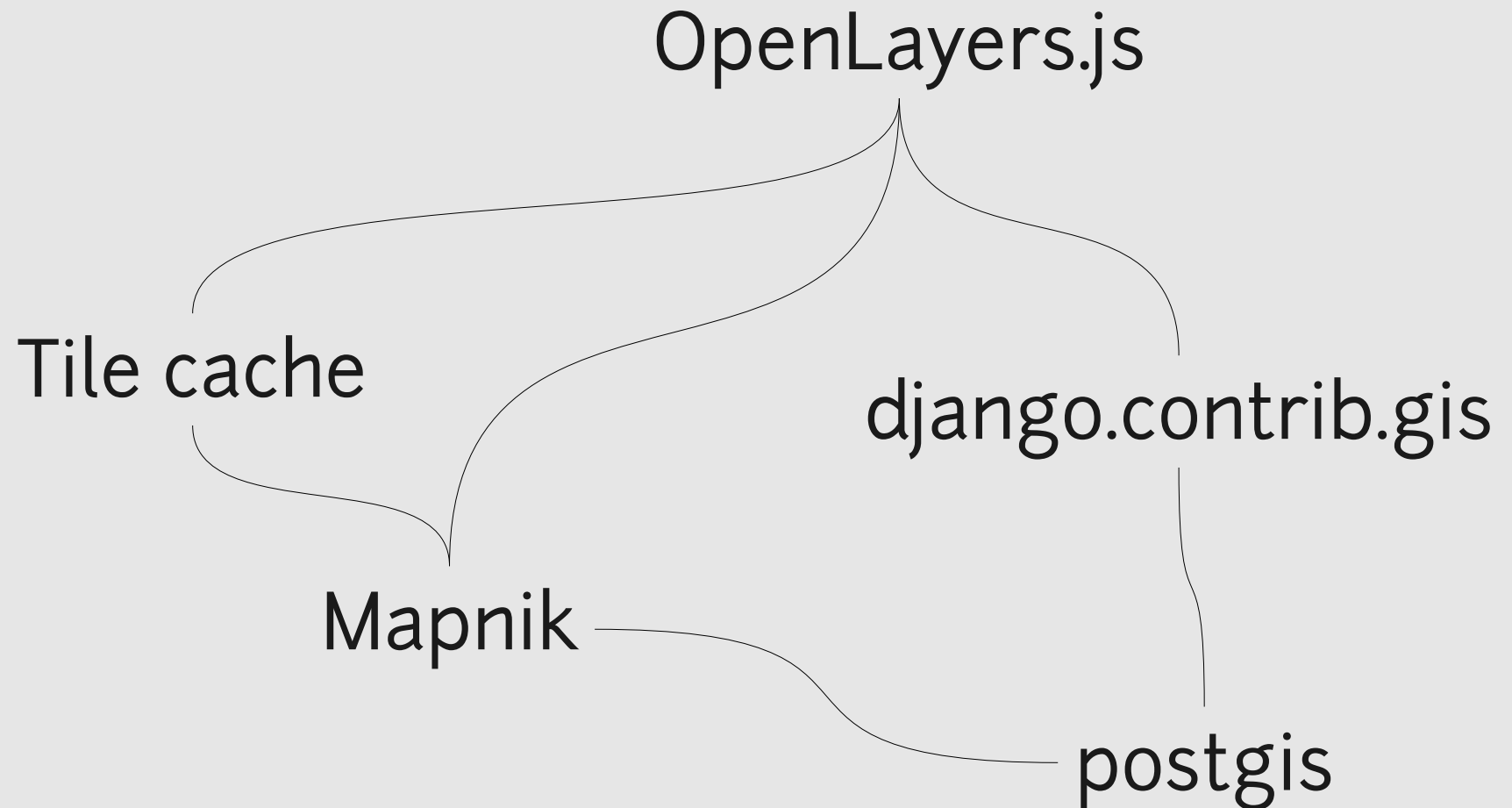
- ✓ Input from raster or vector formats

Tilecache

- ✓ ... or mod_tile or tilestache or other

- ✓ Avoid recomputing common data

GeoDjango

- Use views to provide subset of data

- Easy default output in formats understood by OpenLayers.

The Stack

OpenLayers.js

Tile cache

django.contrib.gis

Mapnik

postgis

Imaginary Maps

- ✔ Need to replace base image

- ✔ GeoAdmin very customisable; easy to do

- ✔ Mapnik WMS server running locally

Code Samples

```python
class Track(models.Model):
    name = models.CharField(unique=True,
        max_length=50)

    path = models.LineStringField(
        geography=True)

    objects = models.GeoManager()

    def __unicode__(self):
        return self.name
```

```python
class AdminBase(admin.GeoModelAdmin):
    openlayers_url = \
        "/static_data/openlayers/OpenLayers.js"
    wms_url = "http://localhost:8001"
    wms_layer = "base"
    wms_name = "Imaginary island"
    default_lat = -0.043409718o624
    default_lon = 150.057492178
```

```python
def _get_request_box(request):

    bbox = request.GET.get("BBOX",
        request.GET.get("bbox"))

    minx, miny, maxx, maxy = [float(elt) for
        elt in bbox.split(",")]

    return geos.Polygon.from_bbox((minx, miny,
        maxx, maxy))
```