TURBOGEARS2 GEOSPATIAL FRAMEWORK

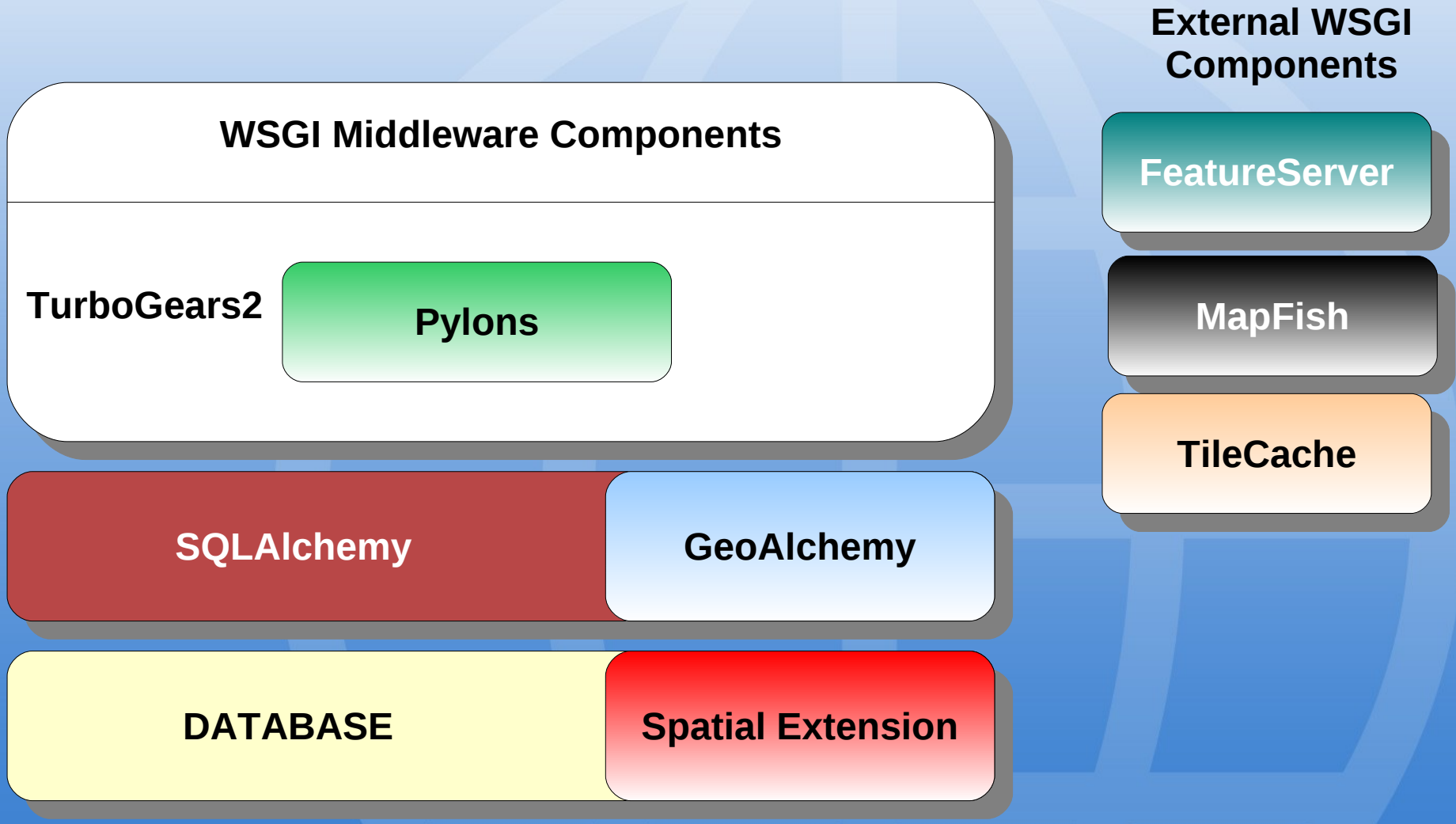
PYCON 2010

SANJIV SINGH
singhsanjivk@gmail.com

Best of Breed Philosophy

- Uses existing GIS libraries (Best of Breed)
  - GeoAlchemy
  - FeatureServer
  - MapFish
  - TileCache
- tgext.geo : TG2 Extension
  - Integrates them into a TG2 App
  - TG2 applications can use GIS
  - GIS applications get a Web Framework

tgext.geo Architecture



GeoAlchemy Model Definition

package/model/road.py

```
from sqlalchemy import *
from geoalchemy import  GeometryColumn, GeometryDDL, LineString

class Road(DeclarativeBase):
    __tablename__ = 'roads'

    id = Column(Integer, primary_key=True)
    name = Column(Unicode(1024), nullable=False)
    geom = GeometryColumn(LineString(2, srid=4326))

GeometryDDL(Road.__table__)
```

Resulting SQL

```
CREATE TABLE roads (
        id INTEGER PRIMARY KEY,
        name VARCHAR(1024) NOT NULL
);

SELECT AddGeometryColumn('', 'roads', 'LINESTRING', 'geom', 2);
```

Spatial Functions

Buffer

```
>>> r = session.query(Road).first()
>>> buffer_geom = session.scalar(
        r.geom.buffer(10.0))
```

Envelope

```
>>> r = session.query(Road).first()
>>> envelope_geom = session.scalar(
        r.geom.envelope)
```

Convex Hull

```
>>> r = session.query(Road).first()
>>> cv_geom = session.scalar(
        r.geom.convex_hull)
```

Spatial Filtering

Intersects

```
>>> r = session.query(Road).first()
>>> res = session.query(Road).filter(
        Road.geom.intersects(r.geom)).all()
```

Touches

```
>>> res = session.query(Lake).filter(
        Lake.geom.touches(r.geom)).all()
```

CoveredBy

```
>>> res = session.query(Road).filter(
        Road.geom.covered_by(l.geom)).all()
```

demo at http://geo.turbogears.org/geoalchemydemo/

FeatureServer

- Restful GIS server
- Supports various Data input formats using the DataSource Interface
  - e,g, Shapefile, PostGIS, DBM, external WFS, etc.
  - Support for GeoAlchemy and AppEngine now available
- Exposes spatial data in many formats
  - e.g. GeoJSON, GML, KML, GeoRSS, etc.
- More details at http://featureserver.org/

Configuring FeatureServer

TurboGears2 config system used for passing params to FeatureServer

```
geo.road.model = package.model
geo.road.class = Road
geo.road.srid = 4326
geo.road.geom_cls = Exterior
geo.road.geom_rel = the_geom
geo.road.join_condition =
Road.id==Exterior.road_id
```

FeatureServer as TG2 Controller

```
from package.model import DBSession
from package.lib.base import BaseController
from repoze.what import predicates
from pylons.i18n import lazy_ugettext as l_
from tgext.geo.featureserver import FeatureServerController

class RootController(BaseController):

    allow_only = predicates.has_permission('road',
        Msg = l_('Restricted to “road” permission')
    road = FeatureServerController('roads', DBSession,
            allow_only)
```

TileCache

- Generating Rich Maps Expensive
- Maps from external source cost Bandwidth
- TileCache:
  - Caches map tiles
  - Supports pre-generated tiles

- Supports several cache storages
  - e.g. Disk, Memcache, Amazon S3, etc.
- More details on http://tilecache.org/

Configuring TileCache

```
[cache]
type = Disk
base = /path/to/disk/cache

[wms_layer]
type = WMS
url = http://labs.metacarta.com/wms/vmap0

[osm_layer]
type = Mapnik
mapfile = /home/user/mapnik/osm.xml
spherical_mercator = true
metatile = yes
bbox = -20037508.34,-20037508.34,20037508.34,20037508.34
resolutions = 156543.0,..........., 0.298582077026
srs = EPSG:900913
```

TileCache as TG2 Controller

```
from package.lib.base import BaseController
from repoze.what import predicates
from pylons.i18n import lazy_ugettext as l_
from tgext.wsgiapps import TileCacheController

class RootController(BaseController):

    allow_only = predicates.has_permission('maps',
        Msg = l_('Restricted to “maps” permission')
    road = TileCacheController(allow_only,
                '/etc/tilecache.cfg')
```

tw.openlayers

- Toscawidget Library for OpenLayers
- Javascript Mapping Toolkit
  - Interactive Maps
  - Layers from different sources
  - e.g. Google, Yahoo, WMS Servers
  - Sophisticated Controls
- BSD License
- More details on http://openlayers.org/

Creating OpenLayers Map Widget

```
from tw.api import WidgetsList
from tw.openlayers import WMS, GML


class MyLayers(WidgetsList):
    ol = WMS(name="OpenLayers WMS",
             url=['http://....
                  labs.metacarta.com/wms/vmap0'],
             options={'layers': 'basic'})
    roads = GML(name="Roads",
             url=['/roads'],
             options={})
```

Creating OpenLayers Map Widget

```
from tw.api import WidgetsList
from tw.openlayers import LayerSwitcher,
         Navigation, PanZoomBar
from tw.openlayers import Map

class MyControls(WidgetsList):
    ls = LayerSwitcher()
    nav = Navigation()
    pzm = PanZoomBar()

map = Map(id='map', layers=MyLayers(),
         controls=MyControls(),
         centerX=0, centerY=0, zoom=3)
```

Creating OpenLayers Map Widget

Controller

```
@expose()
def index(self):
    tmpl_context.map = map
    return dict()
```

Template

```
${tmpl_context.map}
```

Creating OpenLayers Map Widget

QUESTIONS