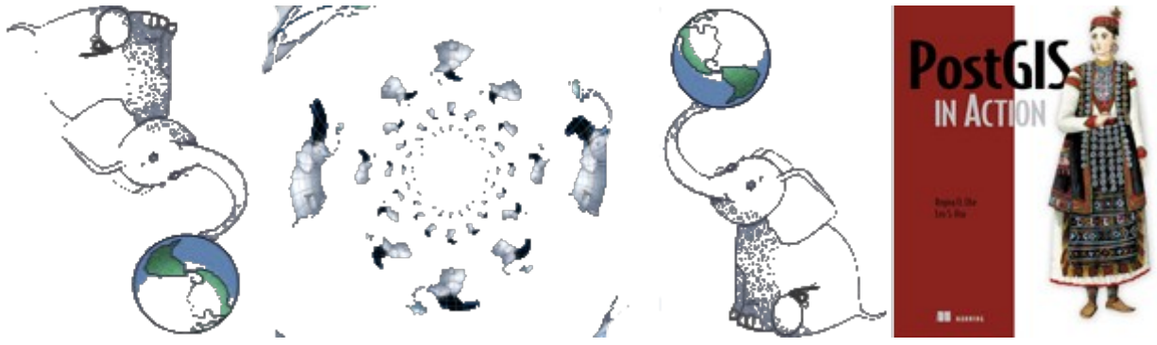
Integrating PostGIS in Web Applications

Leo Hsu
Regina Obe
http://www.paragoncorporation.com



http://www.postgis.us

What we will be covering

- Links and Relevant Ecosystem of Open Source GIS and OpenGIS
- Glossary of Terms
- Tools to Load/Extract spatial data from PostGIS
- Proximity Queries with PostGIS 1.5+
- OpenLayers
- Using OpenStreetMap as a base layer on a map
- Using WMS Layers in OpenLayers
- Geoserver and MapServer
- Publishing PostGIS data with WMS
- Publishing PostGIS data with WFS
- Roll your own custom layers using PostGIS output functions
- Enhancing OpenLayers with GeoExt

Links and Relevant Ecosystem of Open Source GIS

- http://www.postgis.org PostGIS Website: (its now an OSGEO incubation project)
- http://trac.osgeo.org/postgis PostGIS community and bug tracker. Also contains user contributed code snippets, tips, and links.
- http://www.osgeo.org/ OSGEO – Open Source Geospatial Foundation, provides hosting and other resources to the Open Source GIS community and OSGEO projects.
- http://postgisonline.org PostGIS Online – work in progress started by Nicklas Avén, one of the core PostGIS developers. It allows you to experiment with PostGIS spatial queries in a database sandbox. Uses Mapserver for displaying query on a map. Also has some step by step tutorials.
- http://openlayers.org OpenLayers – Javascript Framework we'll be demonstrating. Was started by MetaCarta, a local Cambridge,MA company and later became an OSGEO project. BSD style license.
- http://geoserver.org GeoServer – a Java Servlet written web mapping server we'll be covering and is also an OSGEO incubation project. GPL licensed.
- http://www.mapserver.org/ MapServer - a C written web mapping server. The first OSGEO project and also the first mapping server to support PostGIS. MIT style licensed.
- http://www.codeplex.com/SharpMap - both a web and desktop mapping toolkit written in C#. Used often to create ASP.Net and more recently Silverlight apps. PostGIS was one of the first data sources it supported. LGPL licensed.
- http://geoext.org – GeoExt is a mapping javascript toolkit that combines the power of OpenLayers for mapping and ExtJs for general Rich Internet Application (RIA) building. Its BSD licensed.
- http://www.extjs.com – A very popular javascript framework for building RIA. Comes packaged with various graphical widgets that can be stretched, moved on screen etc.
- http://www.opengeospatial.org/ Open Geospatial Consortium (OGC) standards making body for geospatial and location based services. Most of the high end GIS vendors are on the committee as well as many open source GIS members. It tries to improve ease of interoperability between the various tool sets.

OUR SATELLITE SITES

- http://www.postgis.us – PostGIS in Action book site – contains code downloads, eventually demos of what is in book, presentations we've done, useful PostGIS links.

  We are co-authors of the upcoming Manning book, **PostGIS in Action,** due out in hard-copy in Summer 2010. You can purchase now via the Manning Early Access Program (MEAP) and read chapters as we complete/update them. First chapter is a free download. Have almost all the chapters complete and on MEAP.

- http://www.bostongis.com – BostonGIS.com is focused on Open Source and Open Geospatial standard tips and tricks.

- http://www.postgresonline.com – Postgres OnLine Journal is focused on providing a resource for PostgreSQL users and newcomers by providing examples that demonstrate PostgreSQL's unique features, and how to use PostgreSQL effectively. Each edition available as free PDF. We try to release monthly/bi-monthly.

Glossary Of Terms

- **EPSG** – European Petroleum Survey Group – they are the authority for a lot of the spatial reference systems used today. In SRS you'll see their initials prefixed EPSG
- **ESRI Shapefile** – kind of the defacto standard of data transporting for GIS. Pretty much every GIS desktop tool and Open source GIS desktop tool, loaders etc support it. Most vector spatial data you can download is in this format. Created by Environmental Systems Research Institute (ESRI) which is the big bad fish vendor in the GIS world. It has a .shp/.shx to store the geometry part and a dbf to store the regular attribute data.
- **GeoJSON** – http://geojson.org/ Geography Javascript Object Notation. New standard. In OGC draft spec. Combines JSON serialization of attribute and adds to it how geometry objects should be serialized as JSON.
- **GEOS** – Geometry Engine Open Source – the engine used for many of PostGIS advanced functions. C++ based.
- **GML** – Geography Markup Language. A data transport XML format standard of OGC often used to output WFS requests
- **KML** – Keyhole Markup Language – another XML standard that includes stylistic features, popularized by Google, and later adopted as a transport standard by OGC.
- **OGC** – Open Geospatial Consortium – standards body for Geospatial interoparability between products
- **OSGEO** – Open source Geospatial Foundation – incubator of many open source projects including PostGIS and GEOS
- **SRID** – Spatial Reference Identifier – this is the primary key (usually a number) in a spatial reference system catalog of options. It is common terminology for OGC spatial databases.
- **SRS** (ID) – Spatial Reference System. This is similar in concept to spatial database SRID. It defines the coordinate system, projection, and datum of spatial data. Most common ones you'll see are EPSG:4326 (WGS 84 long lat), EPSG:3875 (Web Mercator), EPSG:900913 (Web Mercator old SRS before Web Mercator became a EPSG standard). In fact PostGIS loaded list of SRIDS have just the numbers 4326, 3875, etc.
- **WGS** – World Geodetic System standard for defining geodetic data, ellipsoid and geoid models that is the foundation of many spatial reference system. Long Lat data using a reference ellipsoid. Common one used is in WGS 1984.
- **WFS** – Web Feature Service – another web mapping standard created by OGC for outputting vector and attribute data.
- **WMS** – Web Mapping Service – one of the standards defined by OGC

Open Source Tools to Load Data and Extract Data

shp2pgsql

Command line data loader packaged with PostGIS

Imports standard ESRI Shapefiles and DBFs* into PostGIS geometry/geography

shp2pgsql-gui

GTK based Graphical user interface data loader with more or less same functionality as the command line loader. Also packaged with PostGIS 1.5.0+. Note: Not pre-packaged with all distributions so you may need to compile yourself. Available for windows distributions via StackBuilder.

pgsql2shp

Command line tool packaged with PostGIS that allows you to export spatial tables and ad-hoc queries to ESRI Shape file format

OGR2OGR/GDAL http://fwtools.maptools.org/

Open source tool packaged in FWTools

Imports 20 different vector and flat file formats and various raster formats. Driver currently being developed to load into new PostGIS WKT Raster type.

Spatial Data Integrator http://www.spatialdataintegrator.com/

ETL tool with geospatial capabilities and graphical workbench. It is based on Talend Open Studio.

Geokettle http://sourceforge.net/projects/geokettle/

This is the geospatial equivalent of Pentaho Data Integration (Kettle). Supports out of the box PostGIS, Oracle Spatial, MySQL, ESRI Shapefiles. It can run as a web service.

Where to get Free Data

World

- OpenStreetMap – http://www.openstreetmap.org is a community driven spatial database and map repository of road networks, bike trails, etc. that recently switched to using PostgreSQL for their back-end. It uses public domain data sourced from various governments and non-profits and enhanced by community editors. They have a tool called OSM2pgsql that you can use to load up their data into your PostGIS enabled database - http://wiki.openstreetmap.org/wiki/Osm2pgsql

- Geocommons - http://finder.geocommons.com/ available in various formats often including ESRI Shapefile and KML

United States

- http://www.census.gov/geo/www/tiger
- http://www.data.gov – just recently launched and has both GIS as well as non-GIS data. If you are interested about your own state – check out the map of states that have launched their own open gov site. http://www.data.gov/statedatasites

  Massachusetts, our home state is on the list, and has tons and tons of free GIS data including spectacular satellite imagery available for download or via their numerous web services.

Canada

- GeoBase: http://www.geobase.ca
- Statistics Canada: http://www12.statcan.ca/english/census06/geo/index.cfm
- Natural Resources Canada http://geogratis.cgdi.gc.ca/

Simple Web Integration Proximity Analysis

- PostGIS Geography introduced in 1.5
- GIST index on your geography column
- ST_DWithin function
- Some long lat data
- Supported All core geometry types – (MULTI) POINT, LINESTRING, POLYGON, Geometry Collections – WGS 84 long lat
- Measurements always returned in meters

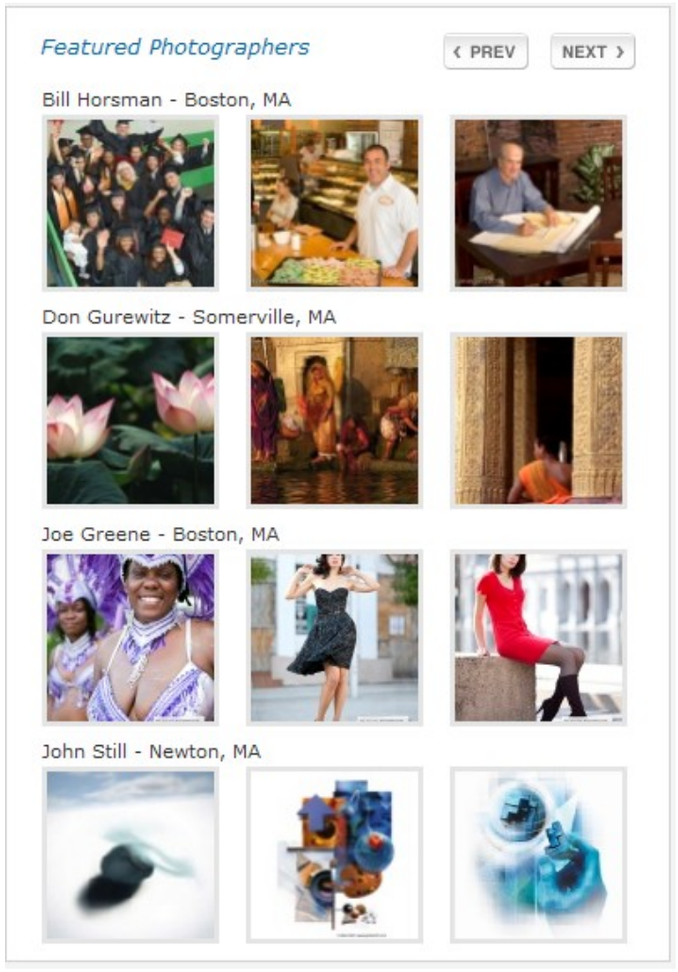
Simple Web Integration
Sample Proximity Query

- What does a geography spatial proximity query look like:

```
$lat = 43.66596;
$lon = -73.13868;
$pt = "ST_GeogFromText('SRID=4326;POINT($lon  $lat)') ";
$miles = 300;
$sql = "SELECT full_name, lfromadd, ltoadd
FROM edges
WHERE ST_DWithin(edges.geog,$pt,1609*$miles)
ORDER BY ST_Distance(edges.geog, $pt) LIMIT 20";
```

- Will return closest 20 streets within 300 miles of a point.

Real World Application
Simple Integration



http://www.photographers.com/
Find the nearest photographers to you

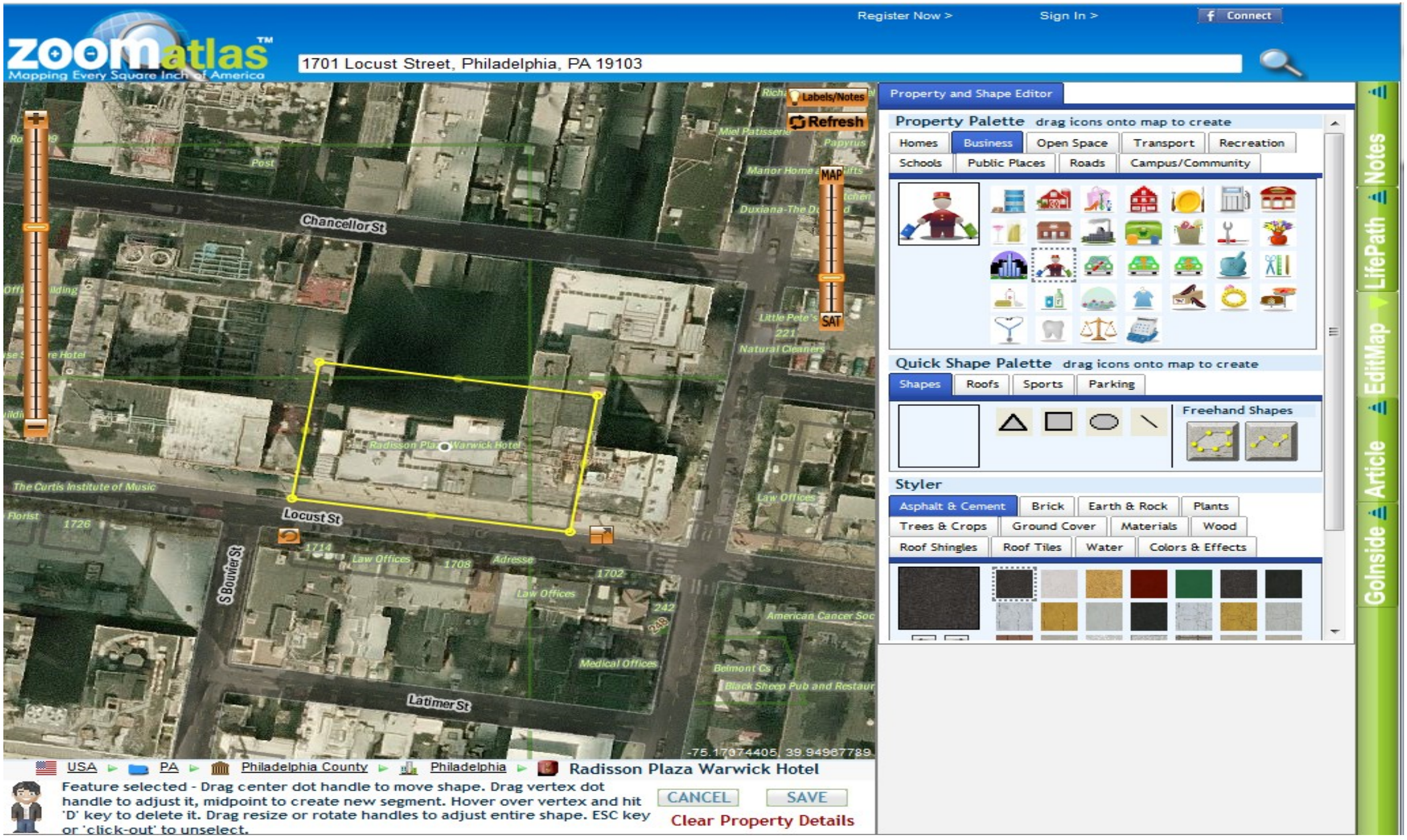
- Uses new PostGIS geography format
- Visitors tagged with GeoIP and shows photographers ordered by GeoIP proximity to visitor

- PostgreSQL backend
- PostGIS geography type to store locations of photographers
- PostgreSQL triggers to keep geography location consistent with input address of photographers.
- PHP and JQuery front end.

OpenLayers JavaScript Framework

- Fairly easy to use Javascript Web Mapping
- Numerous Mapping drivers (OGC – WMS, WFS, WFS-T)
- Supports Proprietary – Google, Bing, Yahoo maps, ArcGIS Rest, ArcGIS IMS, etc.
- OGC transport vector formats – GML, GeoJSON, KML
- You can overlay all on a single map

OpenLayers Highly Customized



Uses PostgreSQL/PostGIS backend for storing vector and user input data. Own customized tile server built with Java. Geoserver for WFS and WFS-T (vector/attribute data editing)

OpenLayers Plain Vanilla
What does it look like?

OpenLayers
What does the code look like?

```
<html>
<head>
    <title>OpenStreetMap Boston</title>
<script src="js/ol28/OpenLayers.js"></script>
<script src="http://www.openstreetmap.org/openlayers/OpenStreetMap.js"></script>
<script src="http://api.maps.yahoo.com/ajaxymap?v=3.0&appid=postgisus"></script>
<script src='http://dev.virtualearth.net/mapcontrol/mapcontrol.ashx?v=6.2'></script>

 <script type="text/javascript">Map setup and init code here</script>
</head>
<body>
<!-- define a DIVs for map and legend -->
    <table>
        <tr>
         <td><div style="width:700; height:600" id="map" /></td>
         <td valign="top"><div id="legend" /></td>
        </tr>
    </table>
    <script type="text/javascript" defer="true">
          init();
    </script>
</body>
</html>
```

OpenLayers
Map Setup Code

```
<script type="text/javascript">
 var lat=42.35815;
 var lon=-71.05971;
 var zoom=15;
 var map; //complex object of type OpenLayers.Map

 function init() {
   map = new OpenLayers.Map ("map", {
     controls:[ new OpenLayers.Control.Navigation(),
       new OpenLayers.Control.PanZoomBar(),new OpenLayers.Control.LayerSwitcher(),
       new OpenLayers.Control.Attribution(),new OpenLayers.Control.MousePosition()
        ],
     maxExtent: new OpenLayers.Bounds(-8879149, 4938750,-7453286, 6017794),
     maxResolution: 156543.0399,
     numZoomLevels: 20,
     units: 'm',
     projection: new OpenLayers.Projection("EPSG:900913"),
     displayProjection: new OpenLayers.Projection("EPSG:4326")
   } );

   var lyrMapnik = new OpenLayers.Layer.OSM.Mapnik("OSM Mapnik");

   var lyryahoohyb = new OpenLayers.Layer.Yahoo(
     "Yahoo Hybrid",
     {'type': YAHOO_MAP_HYB, 'sphericalMercator': true}
   );

   var lyrbing = new OpenLayers.Layer.VirtualEarth(
     "Bing",
     {'type': VEMapStyle.Hybrid, 'sphericalMercator': true}
   );

   map.addLayers([lyrMapnik, lyryahoohyb, lyrbing])

   if( ! map.getCenter() ){
     var lonLat = new OpenLayers.LonLat(lon, lat).transform(new OpenLayers.Projection("EPSG:4326"), map.getProjectionObject());
     map.setCenter (lonLat, zoom);
   }
 }

</script>
```

Web Mapping Service (WMS)

- Dishes out images of requested sections of geographic data as images (RASTER) even when data is of a vector nature
- Many commercial ones
- Many Open source ones – Some common ones that work with PostGIS
  - MapServer - http://mapserver.org/ (Written in C. Most often run as a CGI)
  - GeoServer - http://geoserver.org (Written in Java/Java Servlets)
  - SharpMap.NET - http://sharpmap.codeplex.com/documentation
    (Written in C#. It has WMS 1.3 capabilities but is more designed for more intimate integration with .NET Framework and Silverlight)

- Data sources generally supported by WMS
  - RASTER – e.g. Aerial photos (JPEG,GeoTiff, MrSID etc.) – sometimes stored in database
  - Vector data Flat file -- e.g ESRI Shapefile, MapInfo tab, and others
  - Vector data OGC Spatial Enabled Database –
    - PostgreSQL/PostGIS
    - SQL Server 2008
    - SpatiaLite (the spatially enabled SQLite which also uses GEOS similar to PostGIS)
    - ESRI ArcSDE (PostgreSQL SDE, SQL Server SDE,Oracle SDE)
    - Oracle Locator/Spatial
    - DB2 or Informix Spatial DataBlade
- Cough cough – even sometimes MySQL

Web Mapping Service (WMS)
What makes up a call

- MassGIS site explaining these (in our Hometown) –
  http://lyceum.massgis.state.ma.us/wiki/doku.php
- Latest WMS standard is 1.3, but most tools only support 1.1.1
- Three main kinds of WMS calls: REQUEST=type of request
  - GetCapabilities – Provides a catalog of what map layers the service offers
    and what projections supported in XML format.
    There are sometimes required SERVICE and VERSION arguments. These are not required
    for services that only support one protocol or version.
    http://giswebservices.massgis.state.ma.us/geoserver/wms?REQUEST=GetCapabilities&VERSION=1.1.1

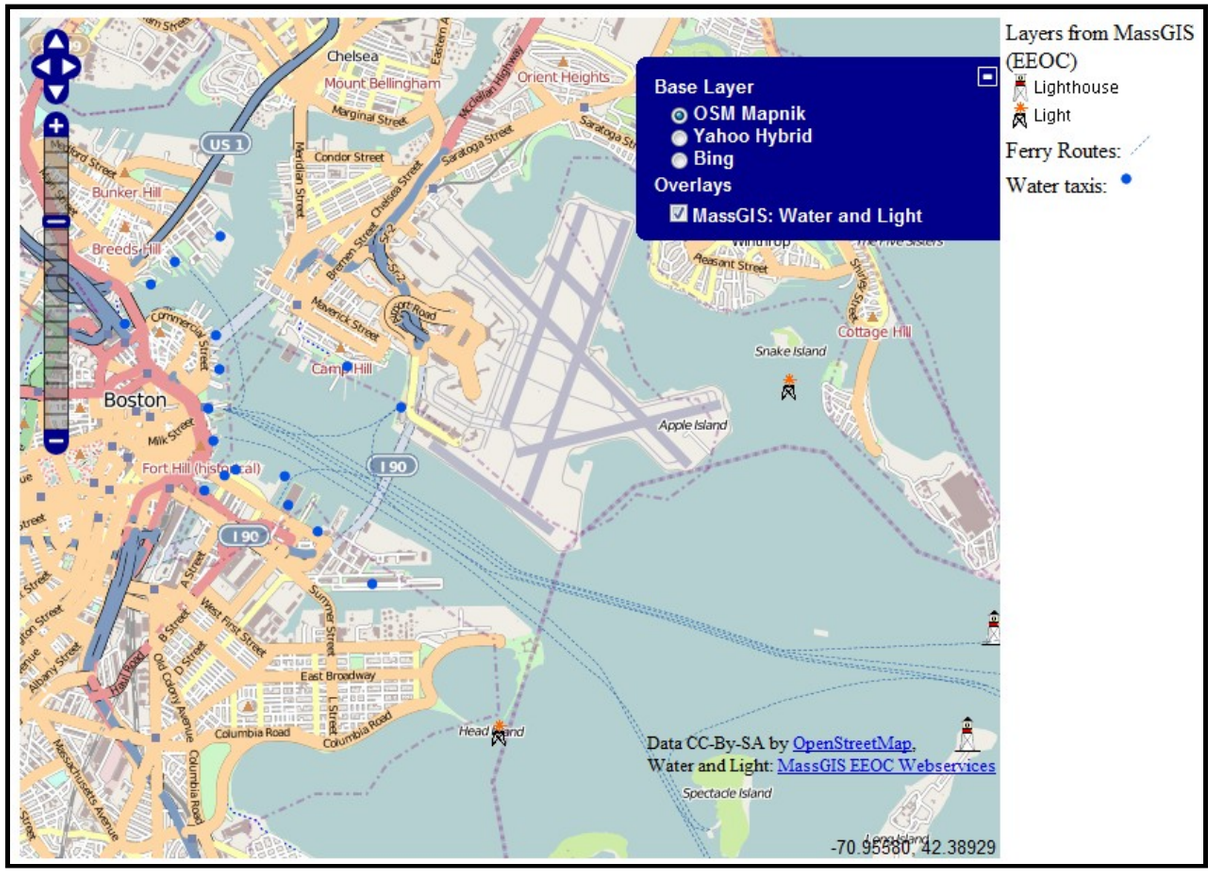
  - GetMap – Dishes out an image of a particular region
    - BBOX – bounding box e.g: `BBOX=-8057798.50,5106032.31,-7765426.87,5319750.24`
    - SRS – the spatial reference system (which in 1.3 is CRS) – `SRS=EPSG:900913` (old web mercator) – new equivalent would be SRS=EPSG:3785
    - SERVICE/VERSION (if service provides more than one) – `SERVICE=WMS&VERSION=1.1.1`
    - LAYERS – a comma separated list of layers to show
    - FORMAT – image format – `FORMAT=image/png`
    - WIDTH – width of tile to return in pixels `WIDTH=500`
    - HEIGHT – height of tile to return in pixels `HEIGHT=500`
    - STYLES – A comma separated list of names of styles supported by the layers selected
    - SLD – URL to a style layer descriptor for more advanced styling
    - SLD_BODY – a fully xml style layer descriptor document

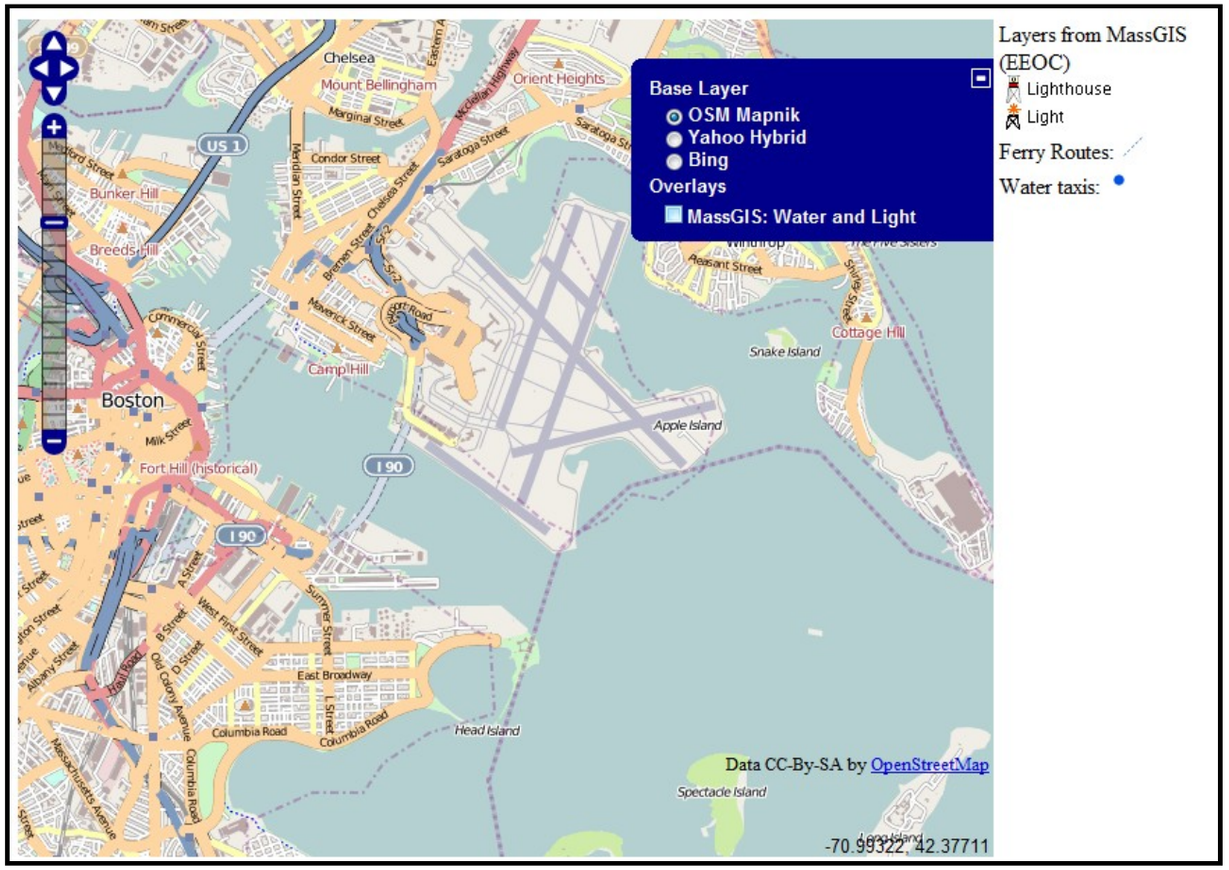
- GetFeatureInfo – usually plain text or GML describing a particular feature. Used to display info about an item when someone clicks on a map. WFS can be used instead too.
- GetLegendGraphic – call to get legend image to show for a layer
  - LAYER
  - FORMAT
  - WIDTH
  - HEIGHT

Web Mapping Service (WMS) Adding a WMS to OpenLayers

```
<script type="text/javascript">
var lat=42.3572;
var lon=-71.01365;
var zoom=13;
 var map; //complex object of type OpenLayers.Map

 function init() {
   map = new OpenLayers.Map ("map", {
     controls:[ new OpenLayers.Control.Navigation(),
       new OpenLayers.Control.PanZoomBar(),new OpenLayers.Control.LayerSwitcher(),
       new OpenLayers.Control.Attribution(), new OpenLayers.Control.MousePosition()
        ],
     maxExtent: new OpenLayers.Bounds(-8879149, 4938750,-7453286, 6017794),
     maxResolution: 156543.0399,
     numZoomLevels: 20,
     units: 'm',
     projection: new OpenLayers.Projection("EPSG:900913"),
     displayProjection: new OpenLayers.Projection("EPSG:4326")
   } );

   var lyrMapnik = new OpenLayers.Layer.OSM.Mapnik("OSM Mapnik");

   var lyryahoohyb = new OpenLayers.Layer.Yahoo(
     "Yahoo Hybrid",
     {'type': YAHOO_MAP_HYB, 'sphericalMercator': true}
   );

    var lyrbing = new OpenLayers.Layer.VirtualEarth(
      "Bing",
      {'type': VEMapStyle.Hybrid, 'sphericalMercator': true}
    );

    map.addLayers([lyrMapnik, lyryahoohyb, lyrbing])

    map.addLayer(new OpenLayers.Layer.WMS("MassGIS: Water and Light", "http://giswebservices.massgis.state.ma.us/geoserver/wms",
      { 'layers': "massgis:GISDATA.LIGHTHOUSES_PT,massgis:GISDATA.FERRYROUTES_ARC,massgis:GISDATA.WATERTAXISTOPS_PT",
            'transparent': "true", 'FORMAT': "image/png"},
      { 'isBaseLayer': false, 'visibility': true, 'buffer': 1, 'singleTile':false, 'tileSize': new OpenLayers.Size(200,200),

        ,'attribution': '<br />Water and Light <a href="http://lyceum.massgis.state.ma.us/wiki/doku.php">MassGIS EEOC Webservices</a>' })
            );

    $('legend').innerHTML = 'Layers from MassGIS (EEOA) <br /><img src="http://giswebservices.massgis.state.ma.us/geoserver/wms/GetLegendGraphic?
VERSION=1.0.0&FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=massgis:GISDATA.LIGHTHOUSES_PT" />
<br />' +
                           'Ferry Routes: <img src="http://giswebservices.massgis.state.ma.us/geoserver/wms/GetLegendGraphic?
VERSION=1.0.0&FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=massgis:GISDATA.FERRYROUTES_ARC" /><br />' +
                           'Water taxis: <img src="http://giswebservices.massgis.state.ma.us/geoserver/wms/GetLegendGraphic?
VERSION=1.0.0&FORMAT=image/png&WIDTH=20&HEIGHT=20&LAYER=massgis:GISDATA.WATERTAXISTOPS_PT" />';


    if( ! map.getCenter() ){
      var lonLat = new OpenLayers.LonLat(lon, lat).transform(new OpenLayers.Projection("EPSG:4326"), map.getProjectionObject());
      map.setCenter (lonLat, zoom);
    }
  }

 </script>
```

Web Mapping Service (WMS) Adding a WMS to OpenLayers





Web Feature Service (WFS)

- Dishes out vector + attribute data usually in the form of GML, GeoJSON, or KML
- Allows for easier client side styling since vector data and attribute data is past as text.
- Many commercial ones
- Many Open source ones – most commonly used that work with PostGIS
  - MapServer - http://mapserver.org/ (supports plain WFS for readonly querying)
  - GeoServer - http://geoserver.org (supports both WFS and WFS-T (the transactional for editing layers))
  - TinyOWS - http://www.tinyows.org (supports both WFS and WFS-T, but not WMS) requires PostGIS 1.5+
    Runs as CGI similar to MapServer so often used to provide the WFS-T component that MapServer lacks
  - FeatureServer – sort of OGC can understand OGC WFS lingo but has its own more common RESTFul way of doing things. Written in Python. BSD style license. Brought to us by MetaCarta. The same folks that gave us open layers.
- Data sources generally supported by WFS
  - Vector data Flat file -- e.g ESRI Shapefile, MapInfo tab, and others
  - Vector data OGC Spatial Enabled Database –
    - PostgreSQL/PostGIS
    - SQL Server 2008
    - ESRI ArcSDE (PostgreSQL SDE, SQL Server SDE,Oracle SDE)
    - Oracle Locator/Spatial
    - DB2 or Informix Spatial DataBlade
    - Cough cough – even sometimes MySQL
- Main Caveat – usually require a proxy if you are trying to access a WFS server via javascript and WFS not coming from your web domain. HTML5 standards supports cross-domain to make this probably not necessary.

Web Feature Service (WFS)
What makes up a call?

- MassGIS site explains this too
  http://lyceum.massgis.state.ma.us/wiki/doku.php?id=filter_functions
- Latest WFS standard is 1.1.0, but most people are running 1.0.0 - http://www.opengeospatial.org/standards/wfs
- Main difference between WFS and WMS is that WFS allows for more advanced queries using spatial filters and usually returns data in some vector or XML format while WMS is more designed for returning data as images.
- Three main kinds of WFS calls: REQUEST=type of request
  - GetCapabilities – Provides an XML catalog of what spatial layers the service offers
    and spatial reference system supported, and spatial operators, functions, and output formats supports.
    There are sometimes required SERVICE and VERSION arguments. Most services that only support one service or version allow you to get away without specifying this.
    http://giswebservices.massgis.state.ma.us/geoserver/wfs?REQUEST=GetCapabilities&VERSION=1.1.0

  - DescribeFeatureType – Gives you list of attributes (table columns) and data types of a feature type table. Need a type name or list of type names separated by comma.
    http://giswebservices.massgis.state.ma.us/geoserver/wfs?REQUEST=DescribeFeatureType&TypeName=massgis:EOPS.UCR_85_S_V,massgis:GIS

GetFeature – Dishes the data about a particular feature or set of features usually in GML format

- BBOX – bounding box e.g: BBOX=-8057798.50,5106032.31,-7765426.87,5319750.24
- SRS – New in 1.1.0 I think (older versions of WFS you were stuck with whatever the default was)
- SERVICE/VERSION (if service provides more than one) – SERVICE=WFS&VERSION=1.1.0
- TYPENAME – a comma separated list of layers (feature types) to show
- FORMAT – Defaults to GML if not specified
- PROPERTYNAME – (featuretype1/featurecol1,featuretype2/featurecol2), etc. (optional will return all columns if not specified)
- FILTER - Too complex to get into
- MAXFEATURES – max number of features to return

But I want to display my own data

Displaying PostGIS Layers (queries)

Many Options to Choose From

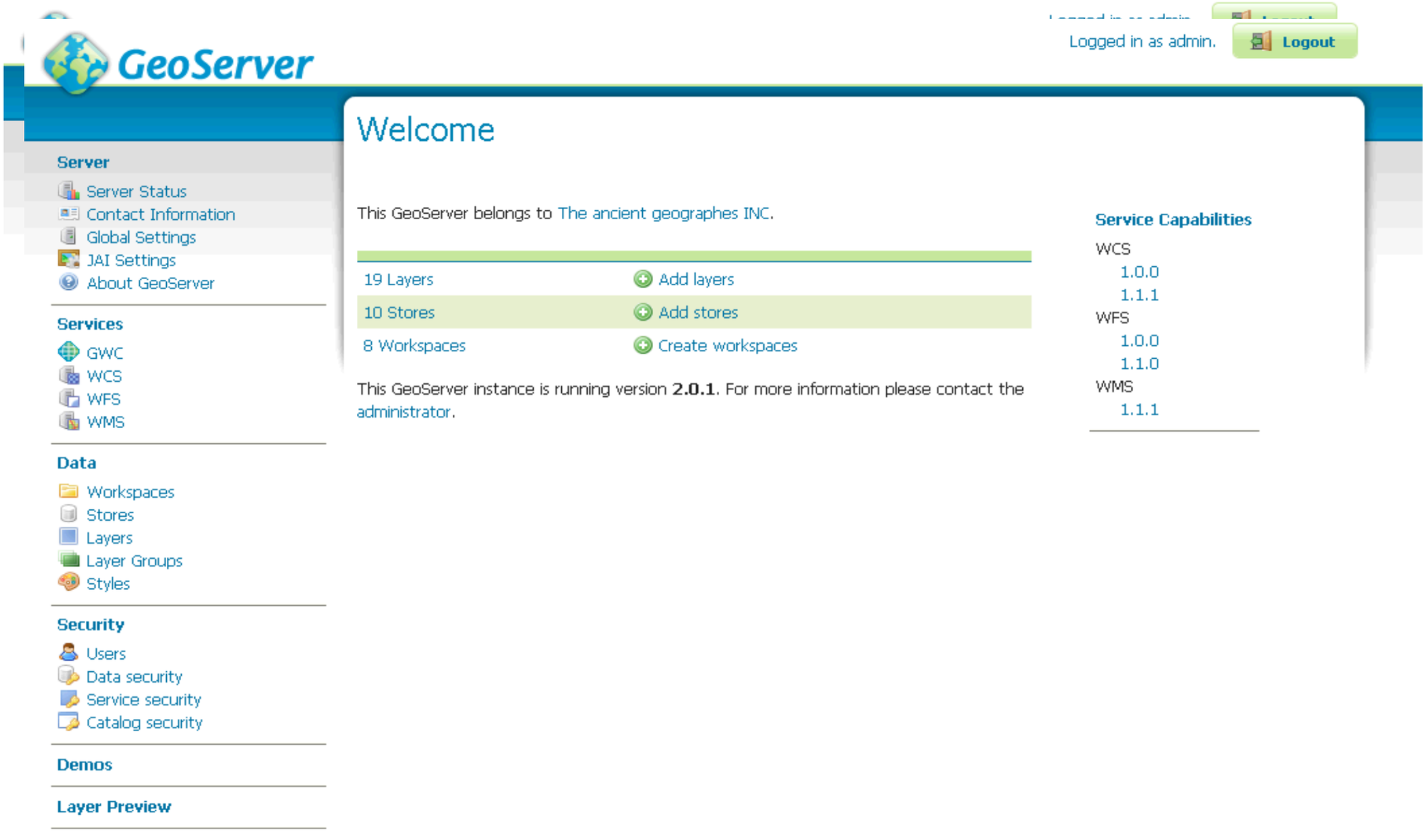
- Roll your own with web scripting language and output functions:
  ST_AsGML, ST_AsGeoJSON, ST_AsKML, ST_AsBinary, ST_AsText
- Toolkits SharpMap.NET, GeoDjango, etc.
- Use Web Mapping Service (WMS) which dishes out image tiles
- Use Web Feature Service (WFS)
  dishes out data in vector form usually as some sort of XML stream
  Usually Geography Markup Language (GML)
- For added speed, a web caching service such as TileCache or GeoWebCache to cache frequently requested tiles reduce roundtrips to database. These sit in front of WMS and generally have a WMS interface.

Geoserver

It is a very functional and fast Mapping server
with A FANCY SMANCY ADMINISTRATIVE CONSOLE.
It is a Ferrari that doesn't cost Ferrari prices.

- Java Servlet based that can run using the packaged Jetty web server
- Latest version 2.0.1
- It can also run under other servers using Tomcat Servlet plugin for Apache or IIS
- Works on most any OS
- GNU GPL licensed Open Source
- Can compile your own or download binaries from http://geoserver.org/display/GEOS/Download
- Great tutorial by Paul Ramsey http://demo.cleverelephant.ca/~pramsey/oss-geostack-intro/doc/en/geoserver.html
- Supports OGC standard interfaces WMS, WFS, WCS
- Supports Common Query Lanaguage (CQL) filters for WMS and WFS
- Supports Style Layer Definitions (SLD) – an OGC XML standard for defining styles (colors to code etc.)
- Also WFS-T (which is lacking in Mapserver)
- Oodles of formats supported can display
- Its WFS can output to not only GML, but also GeoJSON, KML, and ESRI Shapefile
- PostGIS native driver

GeoServer Fancy Smancy Administrative Console

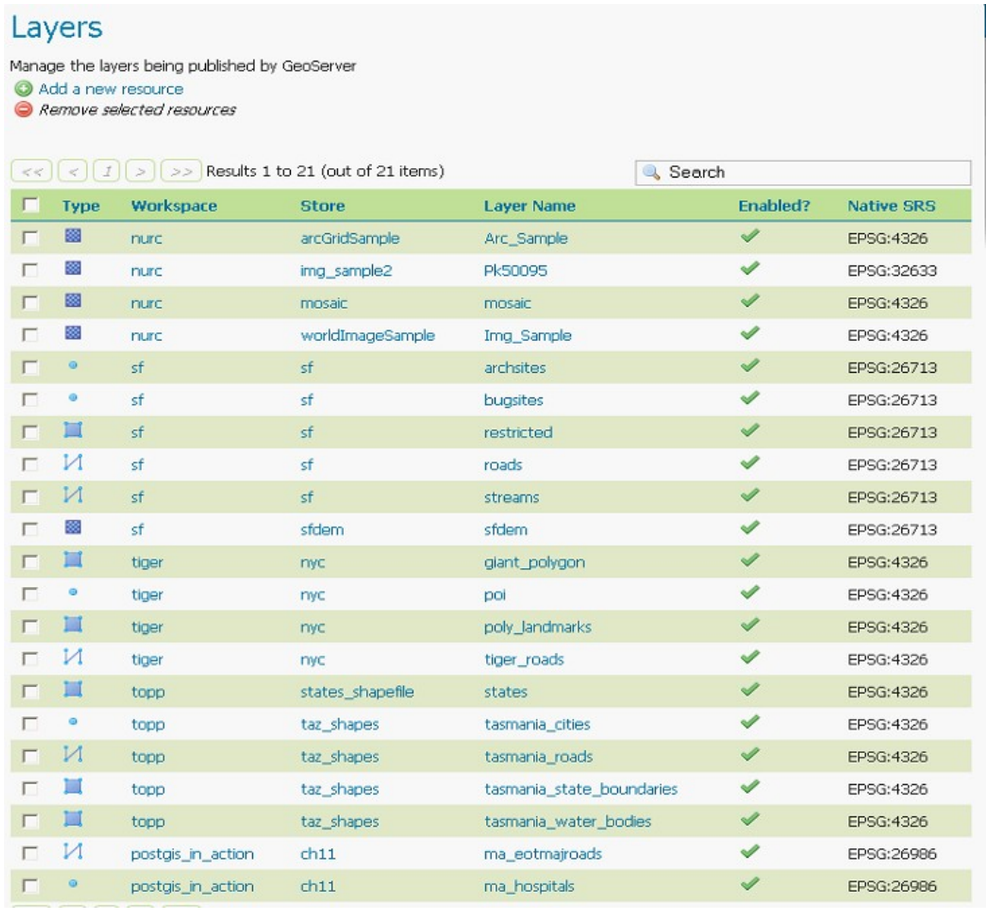


GeoServer: Creating PostGIS Store

1. Create a workspace
2. Add a new data source
3. Choose PostGIS
4. Fill in Info
5. 3-4 For each PostgreSQL schema you have with spatial data

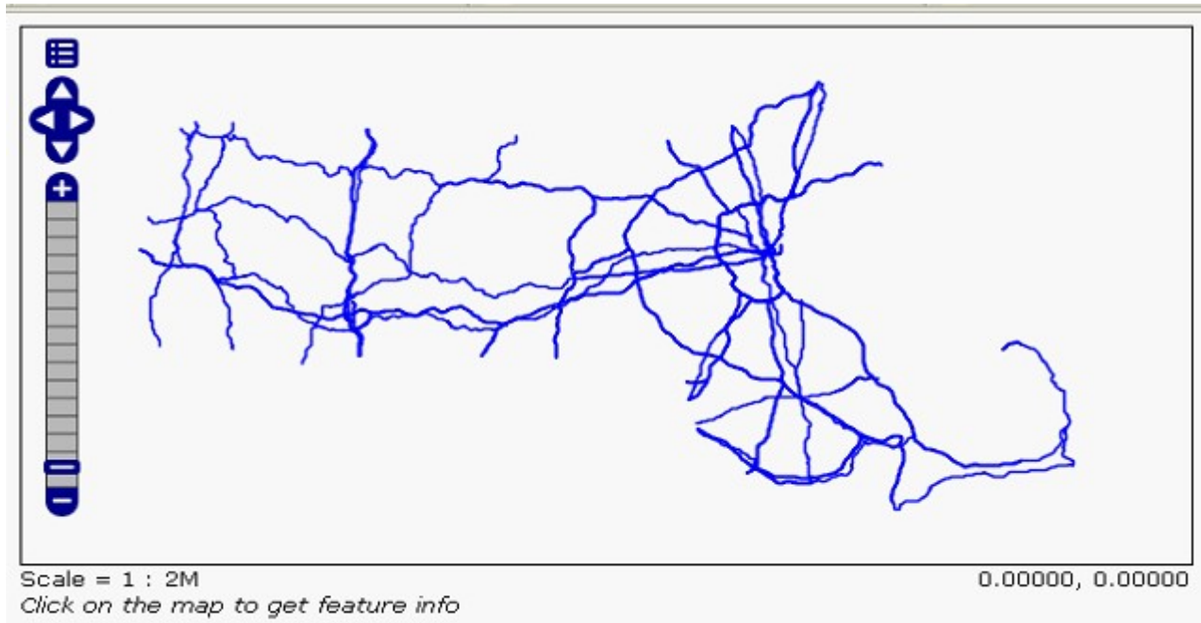
GeoServer: Publishing PostGIS Layers

1. Choose Layers
2. Add a new Resource
3. Select your data store
4. Publish a layer (PostgreSQL Table)
5. Fill in meta data
6. Click compute bounding boxes from data
7. 1-6 Repeat for each layer



GeoServer: See the layers in WMS/WFS

1. Click Layer Preview
2. Many output formats clicking OpenLayers will s
   autogenerated map with your layer



Mapserver – Doesn't need to be hugged

It is a very functional and fast Mapserver
with no fancy smancy administrative console
It is THE Mapserver
Doesn't need a nice comfy bed to lay on like GeoServer
Just throw a mat on the floor

- CGI/FastCGI exe for those who want immediate productivity
- Latest version 5.6.2 just released
- PHP Mapscript / Python MapScript / .NET mapscript and other Mapscripts
  for more granular control but requires you do a lot of the plumbing yourself
- Works on most any OS and most any web server (APACHE,IIS, Light whatever)
- BSD style license
- Can compile your own or download binaries from http://mapserver.org/download.html#binaries
  - If you want to use CSharp Mapscript (which really can be used in any .NET language)
    We suggest FWTools build which contains a csharp folder containing the interop extensions.
- Has its own non OGC standard interface
- Also supports OGC standard interfaces WMS, WFS, WCS etc. including new WMS 1.3.0 standard
- Oodles of formats supported with a pluggable format architecture for registering additional plugins
- PostGIS native driver built-in and in fact the very first mapping server to support PostGIS
- Supports queries as layers
- We'll focus on its OGC functionality

Mapserver

5 Things you should know

For WMS/WFS, all you need to create is a .map file

5 important things to know about map files

1. In mapserver 6.0 – mapfile will support new Mapserver XML format natively
2. Mapserver 5.6+ there is an XML standard and xsd as well as an xsl to convert to standard MapServer map file format
1. If you have password sensitive info in your map, make sure your password info is not accessible via http: Number of ways of doing this
   - Can keep mapfile out of the web root
   - Keep password sensitive info in an include that is never dished out
   - Encrypt passwords with msencrypt executable packaged with mapserver
   - Via your Apache or IIS config – set .map files to not be dished out

1. Need to add wfs/wms/ows meta data options for it to behave like a proper WMS/WFS service
2. You can have INCLUDES pretty much anywhere in the map file.

Mapserver Mapfile for WMS/WFS

```
MAP
  INCLUDE "config.inc.map"
  NAME "POSTGIS_IN_ACTION"  #name to give your map service
  # This is the extent of your map in the srs units of map
  EXTENT 221238 881125 246486 910582
  UNITS meters

  # Default projection of your map - all layers in different
  # projection will be reprojected to this
  PROJECTION
   "init=epsg:26986"
  END
  WEB
    MINSCALEDENOM 100
    MAXSCALEDENOM 100000
    METADATA
      ows_title "PostGIS in Action Chapter 11"
      ows_onlineresource "http://www.postgis.us/demo/chapter_11/GetPAMap.ashx?"
      wms_version "1.1.1"
      wms_srs   "EPSG:2249 EPSG:4326 EPSG:26986 EPSG:3785 EPSG:900913"
      wfs_version "1.0.0"
      wfs_srs   "EPSG:26986"
    END
  END #End Web

  INCLUDE "layers.inc.map"
END # Map File
```

Mapfile Includes

config.inc.map

```
CONFIG PROJ_LIB "/path/to/proj_lib"
SYMBOLSET "symbols/postgis_in_action.sym"
FONTSET "symbols/fonts/fonts.list"
```

postgis.config

```
CONNECTIONTYPE POSTGIS
CONNECTION "host=localhost dbname=somedb user=someuser port=5434 password=somepassword"
PROCESSING "CLOSE_CONNECTION=DEFER"
```

Mapfile PostGIS Layers

```
LAYER
 NAME hospitals
 TYPE POINT
 STATUS ON
 DUMP TRUE
 INCLUDE "postgis.config"
 DATA "geom from ch11.ma_hospitals using unique gid using srid=26986"
 METADATA
  ows_title "Massachusetts Hospitals"
  gml_include_items "all"
  wfs_include_items "all"
  ows_featureid "gid"
 END
 CLASS
  SYMBOL 'circle'
  EXPRESSION ('[trauma]' = 'Y')
  SIZE 10
  COLOR 255 0 0
 END
 CLASS
  SYMBOL 'circle'
  EXPRESSION ('[trauma]' = 'N')
  COLOR 0 200 100
  SIZE 10
 END
```

```
LAYER
 NAME openspace
 TYPE POLYGON
 STATUS ON
 DUMP TRUE
 INCLUDE "postgis.config"
 DATA "geom from ch11.ma_openspace using unique gid using srid=26986"
 PROJECTION
  "init=epsg:26986"
 END
 METADATA
  ows_title "Massachusetts OpenSpace"
  gml_include_items "all"
  ows_featureid "gid"
  wms_srs   "EPSG:2249 EPSG:4269 EPSG:4326 EPSG:26986 EPSG:3785 EPSG:900913"
 END
 OPACITY 70
 CLASS
  COLOR 153 0 102
  OVERLAYCOLOR 102 102 0
  OVERLAYOUTLINECOLOR 200 0 0
 END
END
```

```
LAYER
 NAME major_roads
 TYPE LINE
 STATUS ON
 DUMP TRUE
 INCLUDE "postgis.config"
 DATA "geom from ch11.ma_eotmajroads using unique gid using srid=26986"
 PROJECTION
   "init=epsg:26986"
 END
 LABELITEM "rt_number"
 METADATA
   ows_title "Massachusetts Major Roads"
   gml_include_items "all"
   ows_featureid "gid"
 END
 CLASS
  COLOR 255 0 0
  LABEL
   TYPE truetype
   FONT arial
   MINDISTANCE 50
   POSITION AUTO
   ANGLE AUTO
   SIZE 6
   COLOR 0 0 0
  END
 END
```

```
 LAYER
  NAME hospitals_anot
  TYPE ANNOTATION
  STATUS OFF
  DUMP TRUE
  INCLUDE "postgis.config"
  DATA "geom from ch11.ma_hospitals using unique gid using srid=26986"
  METADATA
   ows_title "Massachusetts Hospitals"
   gml_include_items "all"
   wfs_include_items "all"
   ows_featureid "gid"
  END
  LABELITEM shortname
  CLASS
   COLOR 255 0 0
   LABEL
    TYPE truetype
    FONT arial
    MINDISTANCE 50
    POSITION AUTO
    SIZE 6
    COLOR 0 0 0
   END
  END
 END
```

How do you call Mapserver as a WFS/WMS Service?

1. Need to call the cgi executable with map=physical_path_to_map_file
2. Rest of arguments you pass are pretty much the standard OGC WFS/WMS calls

Examples:
Get Capabilities
http://mydomain/mapservbin/mapserv.exe?map=C:/mapserv/maps/postgis_in_action.map&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetCapabilities

Etc.
Etc.

Wrap your Mapserver in a Reverse Proxy

The ugly and annoying truths about Mapserver

- Map file paths can get very long and ugly, particularly on windows.
- If you do development on one computer and upload to the other have to change all your code to go to absolute paths. Again very annoying.

Alternatives

- Solution – either write a CGI wrapper such as described here http://mapserver.org/cgi/wrapper.html
- Write a regular page or handler in ASP.NET, PHP, *stuff your favorite web language* here that calls your mapserv with the right map file and maybe also WMS and VERSION.

The wrap in a proxy has many advantages over just a CGI wrapper

a) Your mapserv CGI bin doesn't need to be on the same server as your web server, though your web server needs to be able to call it by HTTP.

b) You can use it to call other services – e.g a Government Web service and get around the cross domain scripting issues of HTML4 when calling WFS.

c) You can use it for load balancing. Maps are more intensive to produce than other web pages – if you have millions of hits you may want to redirect to various servers and add more servers as needed.

ASP.NET Reverse Proxy (VB.NET)

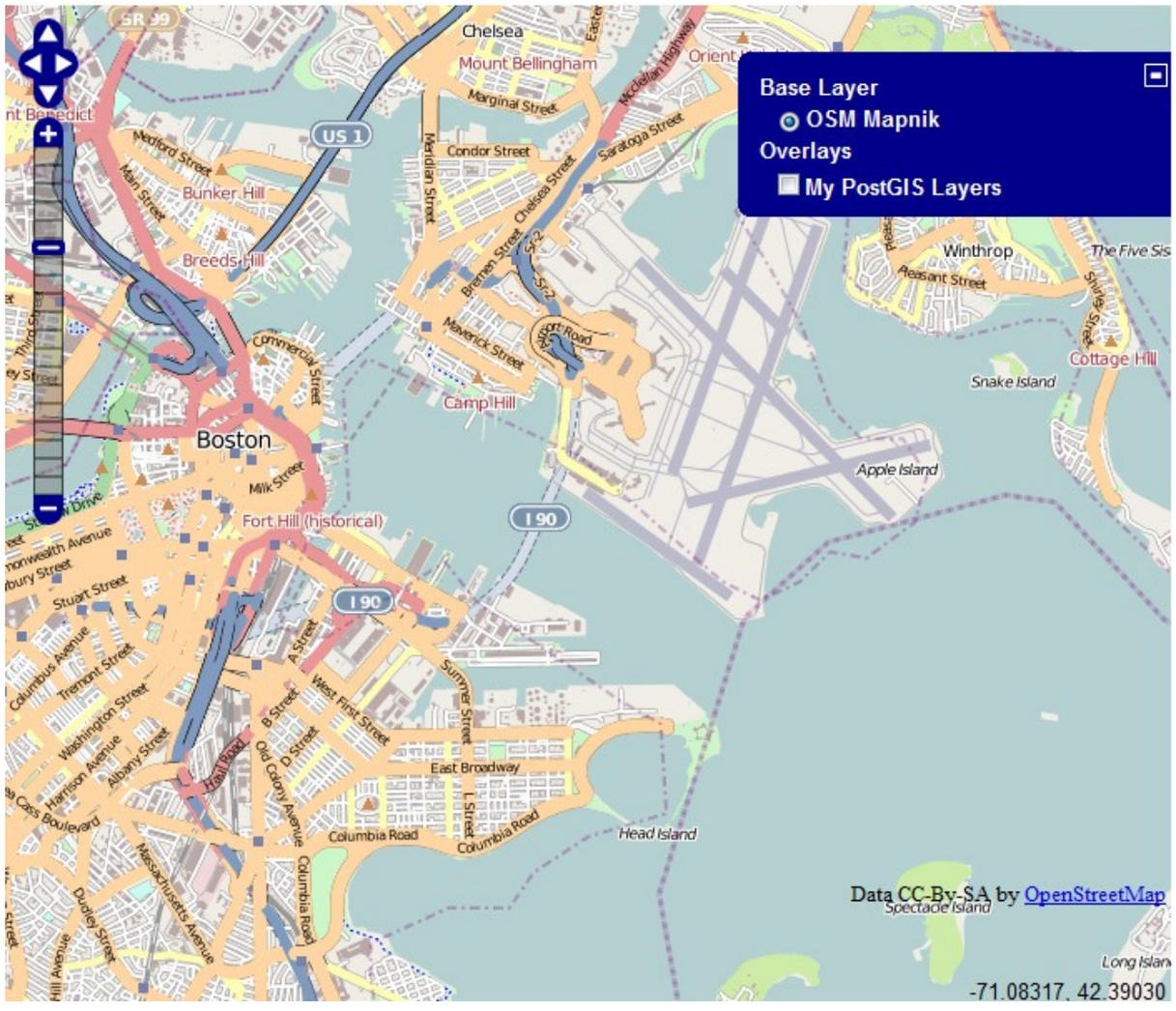
```
<%@ WebHandler Language="VB" Class="GetPAMap" %>
Imports System
Imports System.Web

Public Class GetPAMap : Implements IHttpHandler
    Public Sub ProcessRequest(ByVal context As HttpContext) Implements IHttpHandler.ProcessRequest
        Dim mapURLStub As String = "http://" + context.Request.ServerVariables("HTTP_HOST") + "/mapservbin/mapserv.exe?map="
        Dim mapfile As String = "c:/mapserver/maps/postgis_in_action.map"
        Dim WebRequestObject As System.Net.HttpWebRequest
        Dim sr As System.IO.StreamReader
        Dim WebResponseObject As System.Net.HttpWebResponse
        Dim sb As New System.Text.StringBuilder()
        Dim key As String
        sb.Append(mapURLStub + mapfile)
        Try
            For Each key In context.Request.QueryString.AllKeys
              sb.Append("&" + key + "=" + context.Server.UrlEncode(context.Request.QueryString(key)))
            Next
            WebRequestObject = DirectCast(System.Net.WebRequest.Create(sb.ToString()), System.Net.HttpWebRequest)
            WebRequestObject.Method = "GET"
            WebResponseObject = DirectCast(WebRequestObject.GetResponse(), System.Net.HttpWebResponse)
            If context.Request("REQUEST").ToString.ToLower() = "getcapabilities" _
                        orElse context.Request("REQUEST").ToString.ToLower() = "getfeatureinfo" Then
                sr = New System.IO.StreamReader(WebResponseObject.GetResponseStream)
                context.Response.ContentType = "application/xml"
                context.Response.Write(sr.ReadToEnd())
            Else 'assume an image is being returned
                context.Response.ContentType = context.Request("format").ToString()
                Dim outs As System.IO.Stream = WebRequestObject.GetResponse().GetResponseStream()
                Dim buffer As Byte() = New Byte(&H1000) {}
                Dim read As Integer
                read = outs.Read(buffer, 0, buffer.Length)
                While (read > 0)
                 context.Response.OutputStream.Write(buffer, 0, read)
                 read = outs.Read(buffer, 0, buffer.Length)
                End While
            End If

            Try
                WebResponseObject.Close()
                WebRequestObject.Abort()
            Catch
            End Try

        Catch Ex As Exception 'assume it fails here if the web request fails
            context.Response.Write(Ex.ToString())
        End Try
    End Sub

    Public ReadOnly Property IsReusable() As Boolean Implements IHttpHandler.IsReusable
        Get
            Return True
        End Get
    End Property
End Class
```

ASP.NET Reverse Proxy (C#)

```
<%@ WebHandler Language="C#" Class="GetPAMapCS" %>
using System;
using System.Web;
using System.Net;

public class GetPAMapCS : IHttpHandler
{
   public void ProcessRequest(HttpContext context)
   {
    string mapURLStub = "http://www.postgis.us/mapservbin/mapserv.exe?map=";
    string mapfile = "c:/mapserver/maps/postgis_in_action.map";

    System.Net.HttpWebRequest WebRequestObject;
    System.IO.StreamReader sr;
    System.Net.HttpWebResponse WebResponseObject;
    System.Text.StringBuilder sb = new System.Text.StringBuilder();
    sb.Append(mapURLStub + mapfile);

    try {
        foreach (var key in context.Request.QueryString.AllKeys) {
           sb.Append("&" + key + "=" + context.Request.QueryString[key]);
        }

        WebRequestObject = (System.Net.HttpWebRequest) System.Net.WebRequest.Create(sb.ToString());
        WebRequestObject.Method = "GET";
        WebResponseObject = (System.Net.HttpWebResponse) WebRequestObject.GetResponse();
        if (context.Request["REQUEST"].ToLower() == "getcapabilities" || context.Request["REQUEST"].ToLower() == "getfeatureinfo") {
          //return xml if getcapabilities or getfeatureinfo
          sr = new System.IO.StreamReader(WebResponseObject.GetResponseStream());
          context.Response.ContentType = "application/xml";
          context.Response.Write(sr.ReadToEnd());
        }
        else {
           //assume an image or some other binary object
           context.Response.ContentType = context.Request["format"].ToString();
           System.IO.Stream outs = WebRequestObject.GetResponse().GetResponseStream();
           byte[] buffer = new byte[0x1000]; int read;
           while ((read = outs.Read(buffer, 0, buffer.Length)) > 0){
               context.Response.OutputStream.Write(buffer, 0, read);
           }
        }

        try {
           WebResponseObject.Close();
           WebRequestObject.Abort();
        }
        catch {}
       }
       catch (Exception Ex) {
        //assume it fails here if the web request fails
        context.Response.Write(Ex.ToString());
       }
  }

  public bool IsReusable {
     get { return true; }
  }
}
```

Showing our Layers in OpenLayers

```
var postgiswmsurl = "http://www.postgis.us/demos/chapter_11/GetPAMap.ashx?"
   map.addLayer(new OpenLayers.Layer.WMS("My PostGIS Layers", postgiswmsurl,
     { 'layers': "hospitals,major_roads,openspace",
             'transparent': "true", 'FORMAT': "image/gif"},
     { 'isBaseLayer': false, 'visibility': true, 'buffer': 1, 'singleTile':false, 'tileSize': new OpenLayers.Size(200,200),
             'attribution': 'Data downloaded from <a href="http://www.mass.gov/mgis/">MassGIS</a>'})
     );
```



PostGIS Layers Edit

You want to edit your custom PostGIS Layers via Web

Fewer than for display but still many

- Roll your own with web scripting language and output functions:
  ST_GeomFromGML, ST_GeomFromKML, ST_GeomFromWKB, ST_GeomFromWKT
- Toolkits SharpMap.NET, Java2D (with PostGIS.jar), etc.
- Use Web Feature Service Transactional (WFS-T) – GeoServer, TinyOWS
- Use FeatureServer REST http://featureserver.org/

Latest WMS standard is 1.3

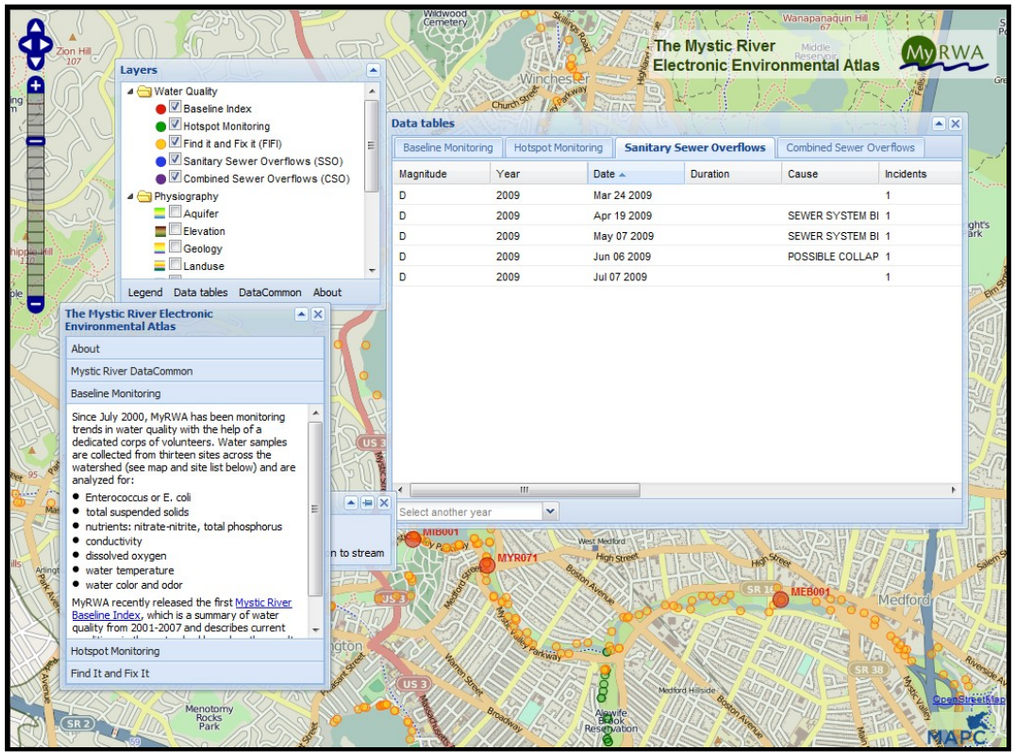
GeoExt
New kid on the block

GeoExt - is a toolkit that extends both OpenLayers and ExtJs to form a new animal that is the Best of both worlds.
http://geoext.org – BSD licensed, but relies on extjs which is GPL v3.
http://openlayers.org – MIT licensed
http://extjs.com – GPL v3

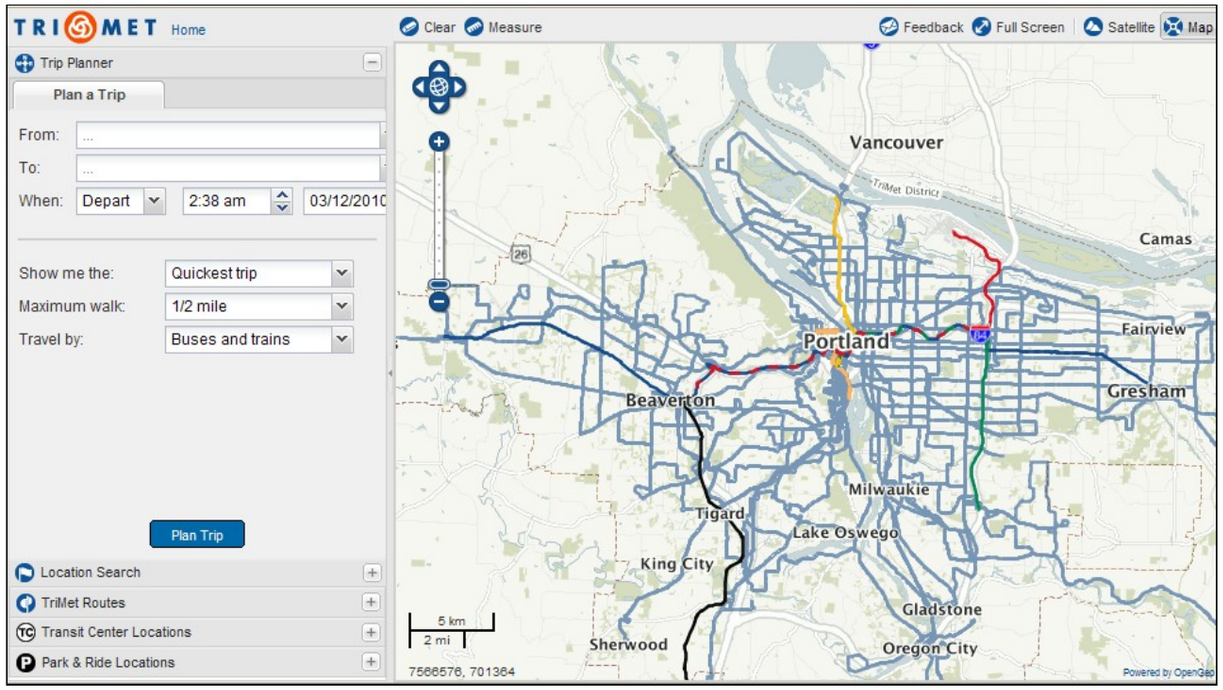
OpenLayers + ExtJS + Glue = GeoExt

Everyone particularly in Open Source GIS community is knocking on the new kid's door.

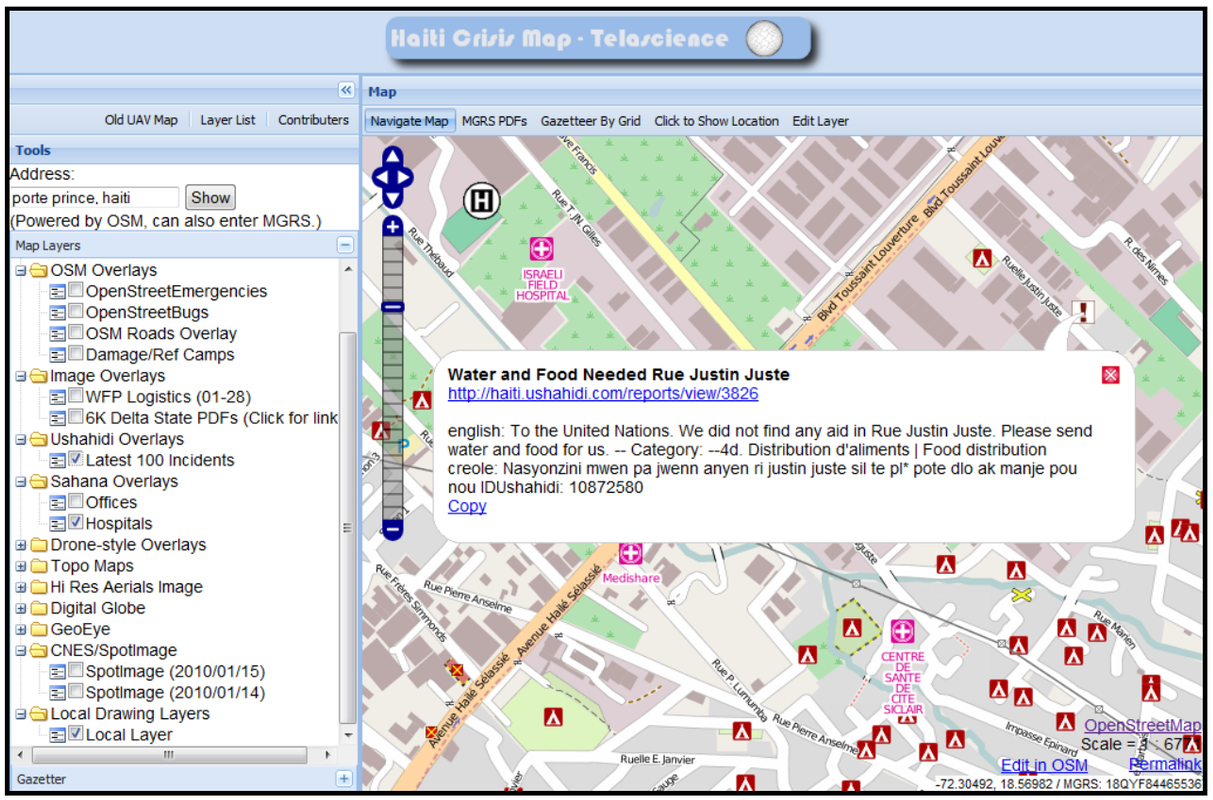
What does GeoExt look like?



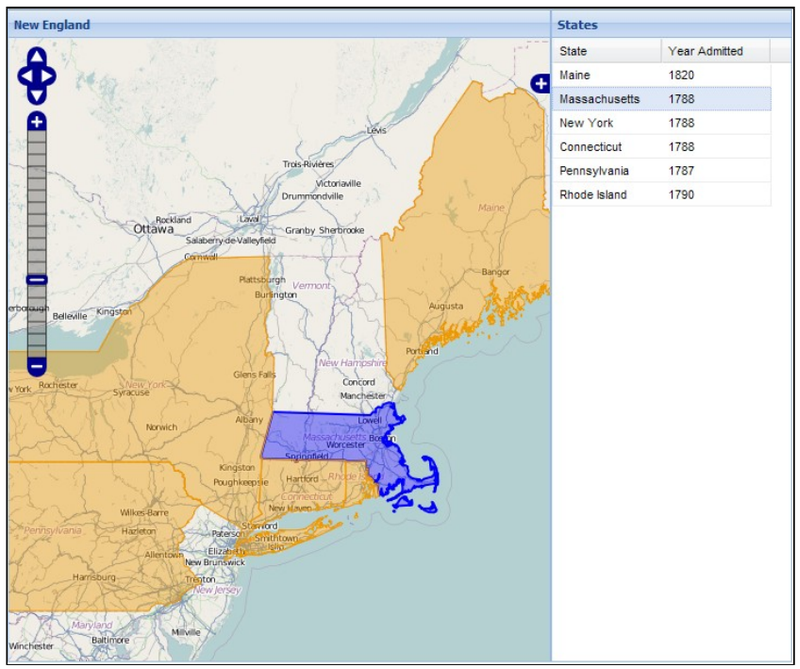
http://maps.mapc.org/myrwa/



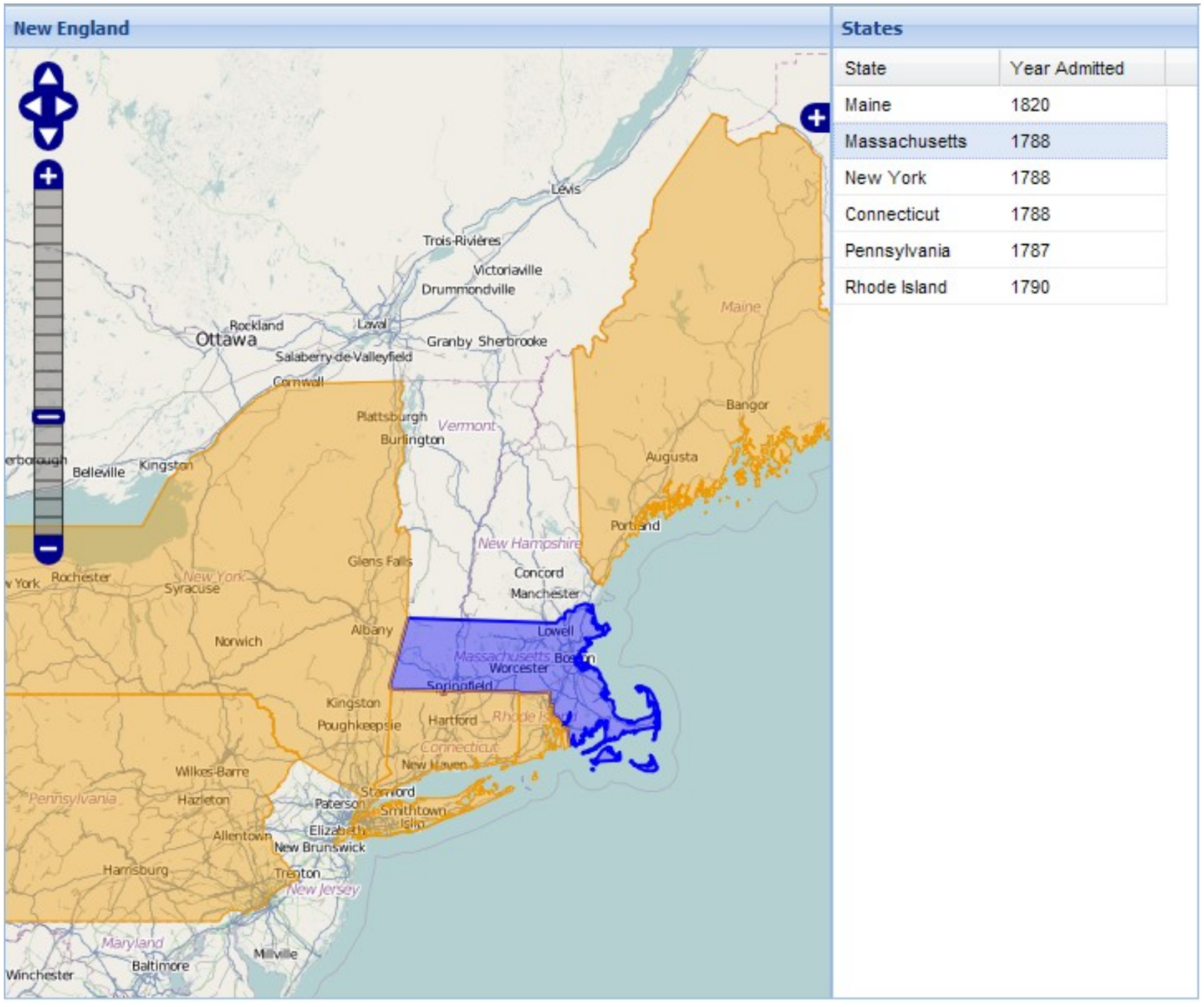
http://ride.trimet.org/?tool=routes



http://haiticrisismap.org/



A boring example to show PostGIS output functions, GeoExt, and Open Street Map 1



A boring example to show PostGIS output functions, GeoExt, and Open Street Map
The htm page

Htm page

```
<html>
<head>
    <title>OpenStreetMap New England States</title>
    <script src="http://api.maps.yahoo.com/ajaxymap?v=3.0&appid=postgisus"></script>
    <script src="js/ol28/OpenLayers.js"></script>
     <script type="text/javascript" src="js/ext-3.1.1/adapter/ext/ext-base.js"></script>
    <script type="text/javascript" src="js/ext-3.1.1/ext-all.js"></script>
    <script src="js/GeoExt.js"></script>
    <script src="http://www.openstreetmap.org/openlayers/OpenStreetMap.js"></script>
    <link rel="stylesheet" type="text/css" href="js/ext-3.1.1/resources/css/ext-all.css" />
<script type="text/javascript" src="geoext_newenglandpanel.js"></script>
</head>
<body>
<!--If we were were using panels and view ports, the divs to hold them would go here-->
</body>
</html>
```

A boring example to show PostGIS output functions, GeoExt, and Open Street Map
The js file

```
var lat=43.66596;
var lon=-73.13868;
var zoom=6;
var map, lyrMapnik, lyryahoo,lyrStates, lonlat, panel, gridPanel;
var prj4326 = new OpenLayers.Projection("EPSG:4326");
var prjmerc = new OpenLayers.Projection("EPSG:900913");
lyrMapnik = new OpenLayers.Layer.OSM.Mapnik("OSM Mapnik");

map = new OpenLayers.Map ( {
    controls:[ new OpenLayers.Control.Navigation(),
      new OpenLayers.Control.PanZoomBar(), new OpenLayers.Control.LayerSwitcher()
       ], maxResolution: 156543.0399, numZoomLevels: 20, units: 'm', projection: prjmerc, displayProjection: prj4326
  } );
lonlat = new OpenLayers.LonLat(lon, lat).transform(prj4326, prjmerc); lyrStates = new OpenLayers.Layer.Vector("States");

Ext.onReady(function() {
   dtstate = new GeoExt.data.FeatureStore({
        layer: lyrStates,
        fields: [
            {name: 'state_name', type: 'string'},
            {name: 'year_adm', type: 'string'}
        ], proxy: new GeoExt.data.ProtocolProxy({
            protocol: new OpenLayers.Protocol.HTTP({
                url: "datafeeder.php?format=json",
                format: new OpenLayers.Format.GeoJSON(), bbox: map.getExtent()
            })
        }), autoLoad: true, setParamsAsOptions: true
    });

    gridPanel = new Ext.grid.GridPanel({
        title: "States", store: dtstate, layout: 'fit', columns: [{
            header: "State",
            dataIndex: "state_name", sortable: true
        }, {header: "Year Admitted",
            dataIndex: "year_adm", sortable: true }],
        sm: new GeoExt.grid.FeatureSelectionModel() });

   panel = new Ext.Panel({
    id:'main-panel',baseCls:'x-plain',renderTo: Ext.getBody(),layout:'table',layoutConfig: {columns:2},
    defaults: {frame:false, height: 600},
    items:[{
      title:'New England', xtype: "gx_mappanel", map: map, layers: [lyrMapnik, lyrStates], zoom: zoom,
      extent: [-8879149, 4938750,-7453286, 6017794], center: lonlat, width: 500},gridPanel]});
});
```

A boring example to show PostGIS output functions, GeoExt, and Open Street Map
The Concept

For this we use the following technologies
- PHP ADODB for our abstraction layer
- PHP Smarty for the GeoJSON, KML template file
- PHP page to bring it all together

A boring example to show PostGIS output functions, GeoExt, and Open Street Map
The PHP Part

```
<?php
include_once("app.inc.php");
class _DataFeeder extends Smarty {

  private $db; //this is our database abstraction layer
  private $debug = false;
  private $supported_formats = array('json'=>'ST_AsGeoJSON', 'kml'=>'ST_AsKML');

  function __construct() {
    $this->plugins_dir = array('plugins', 'extraplugins');

    /** Change default smarty delimeter from { } so we don't have to keep escaping json notation
      and gets treated as html comment by html editors **/
    $this->left_delimiter = '<!--(';
    $this->right_delimiter = ')-->';

    if (!isset($this->db)) {
       $this->db = &ADONewConnection(DSN);
      if (!$this->db) {
        die("Cannot connect to server");
      }
      else {
        $this->db->SetFetchMode(ADODB_FETCH_ASSOC);
        $this->db->debug = $this->debug;
      }
    }
    $this->page_load();

  }
  function page_load(){
    $data_template = 'data_json.tpl';
    if (!empty($_REQUEST['format']) ){
      $format = $_REQUEST['format'];
    }
    $convertfunction = $this->supported_formats[$format];
    if ( empty($convertfunction) ){
      $convertfunction = 'ST_AsGeoJSON';
    }
    else {
      $data_template = "data_$format.tpl";
    }
    $sql = "SELECT state as id, state As state_name,year_adm, $convertfunction(ST_SimplifyPreserveTopology(ST_Transform(the_geom,900913),1000)) As geom
              FROM us.states
              WHERE state IN('Massachusetts', 'Rhode Island', 'Connecticut','Maine', 'New York', 'Pennsylvania')";
    $rsdata = $this->db->Execute($sql)->GetRows();
    $this->assign('rs', $rsdata);
    $this->display($data_template);
  }

}
new _DataFeeder;
?>
```

A boring example to show PostGIS output functions, GeoExt, and Open Street Map
The Smarty templates

data_json.tpl

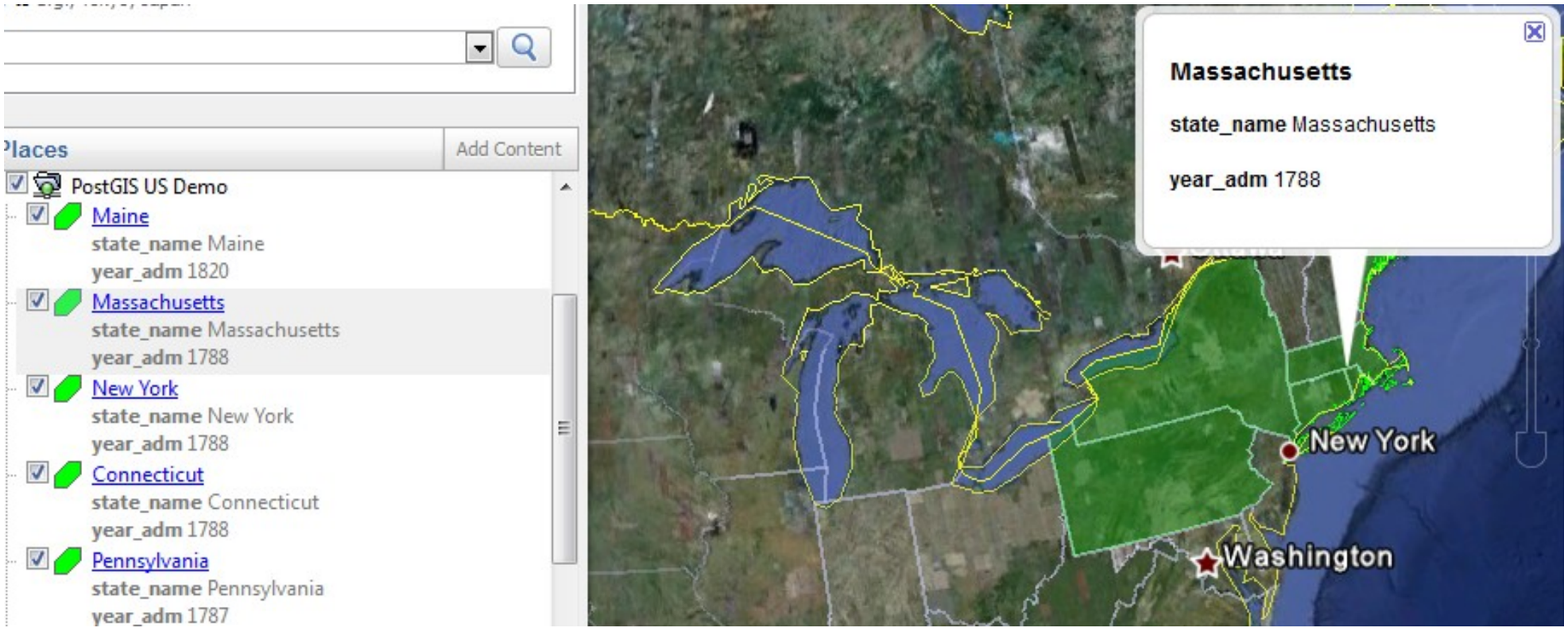
```
{
"type": "FeatureCollection",
"features": [
<!--(section name=sec loop=$rs)-->
        { "type": "Feature", "properties": {"id":<!--($rs[sec].id|json_encode)-->
                <!--(foreach from=$rs[sec] key=prop item=val)-->
                        <!--(if $prop != 'geom' && $prop != 'id')-->,<!--($prop|json_encode)-->:<!--($val|
json_encode)--><!--(/if)-->
                <!--(/foreach)-->
        },
        "geometry": <!--($rs[sec].geom)-->
        }
<!--(if not $smarty.section.sec.last)-->,<!--(/if)-->
<!--(/section)-->
        ]
}
```

data_kml.tpl

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://earth.google.com/kml/2.1'>
<Document>
<Style id='defaultStyle'>
   <LineStyle><color>ff00ff00</color><width>3</width></LineStyle>
   <PolyStyle><color>5f00ff00</color><outline>1</outline></PolyStyle>
</Style>
<!-- load additional styles here -->
<!--(section name=sec loop=$rs)-->
<Placemark>
          <name><!--($rs[sec].id|escape:html)--></name>
<description>
<!--(foreach from=$rs[sec] key=prop item=val)--><!--(if $prop != 'geom' && $prop != 'id')--><b><!--($prop|escape:html)--></b>
<!--($val|escape:html)--><br /><!--(/if)--><!--(/foreach)-->
</description>
   <styleUrl>#defaultStyle</styleUrl>
  <!--($rs[sec].geom)-->
</Placemark>
<!--(/section)-->
</Document>
</kml>
```

A boring example to show PostGIS output functions, Google Earth Viewing using KML our KML output

data_feeder.php?format=kml



Questions