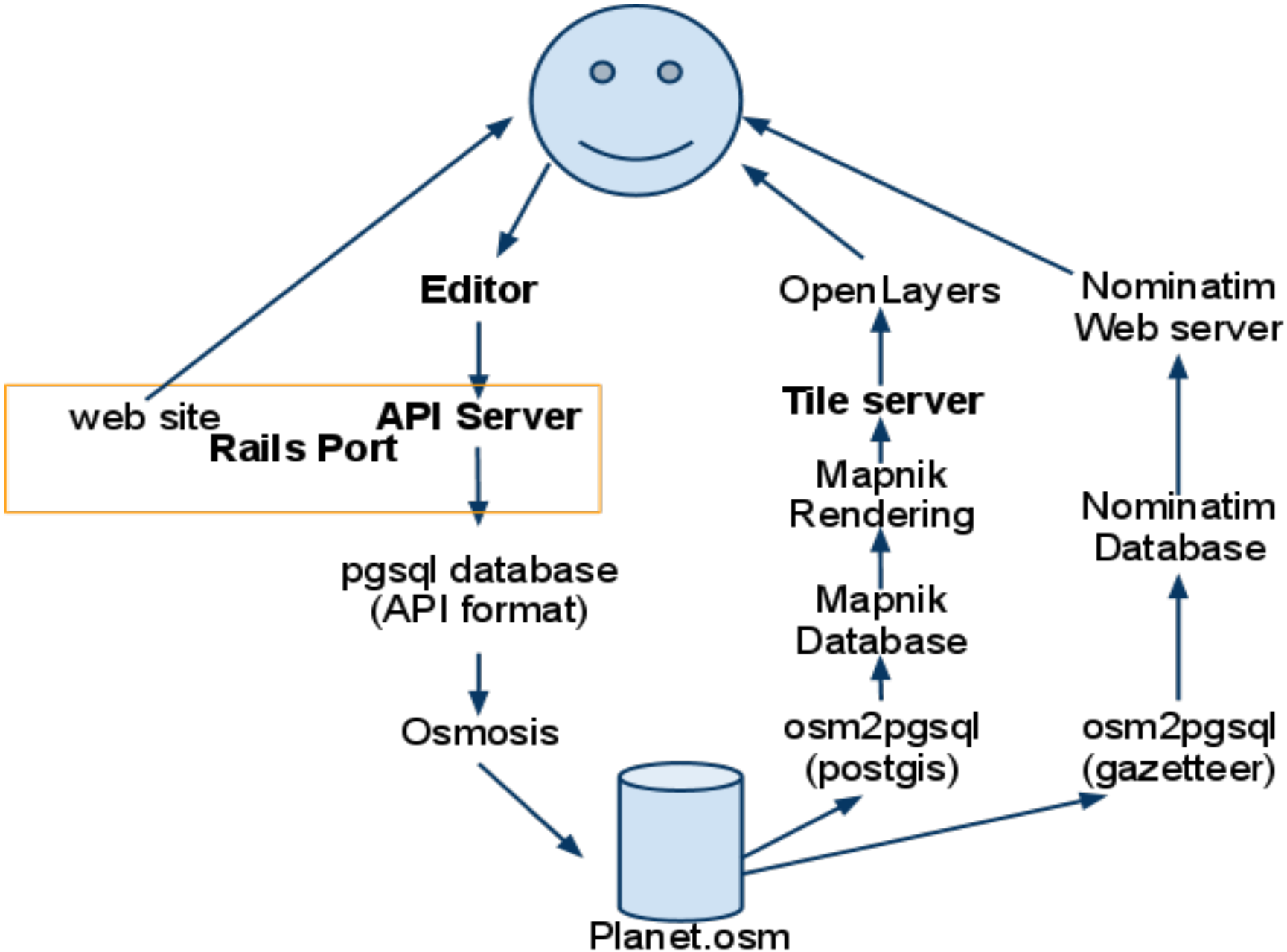
OpenStreetMap technical documentation

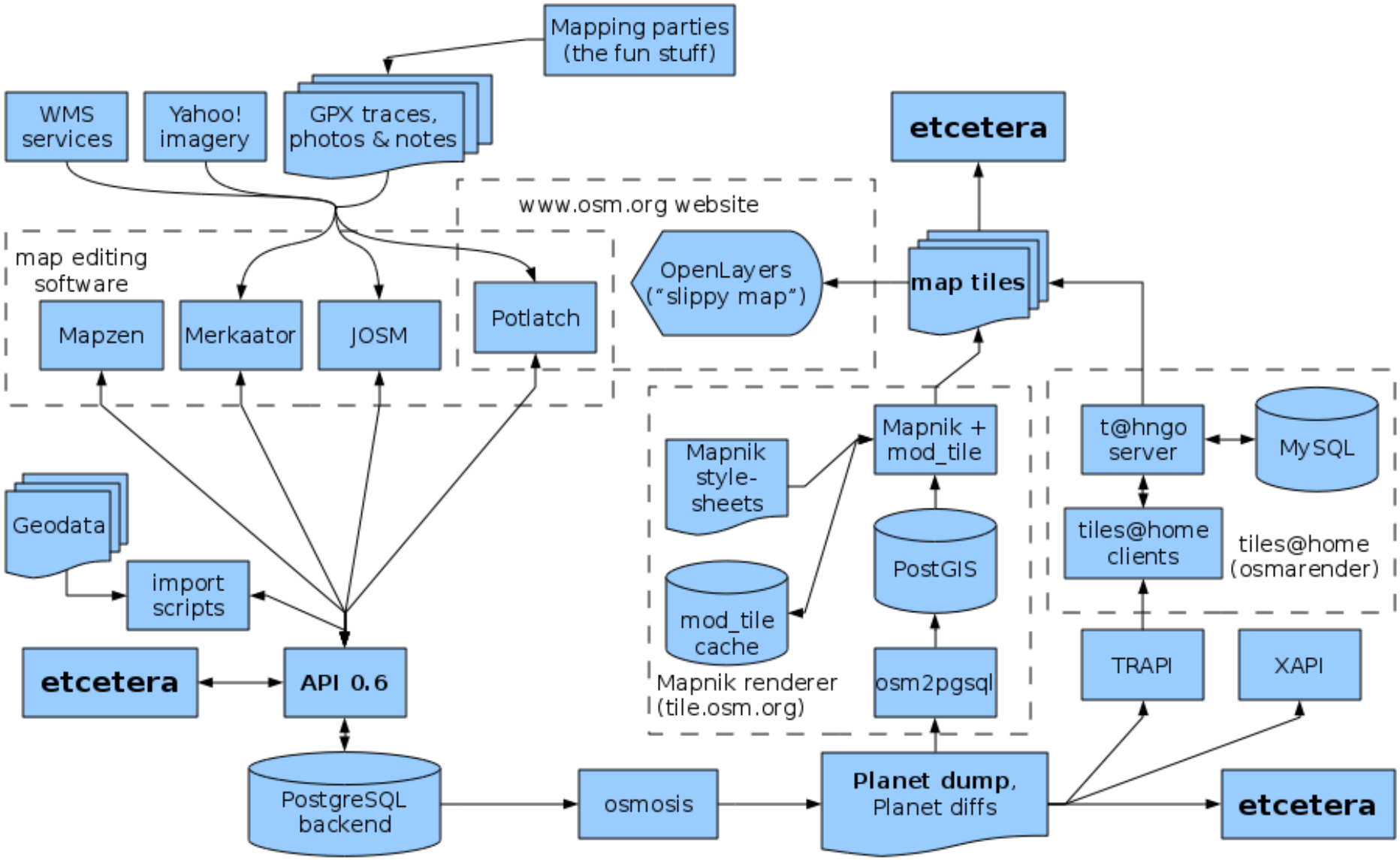
slides to be shared and edited in a collaborative and open manner

OpenStreetMap dataflow



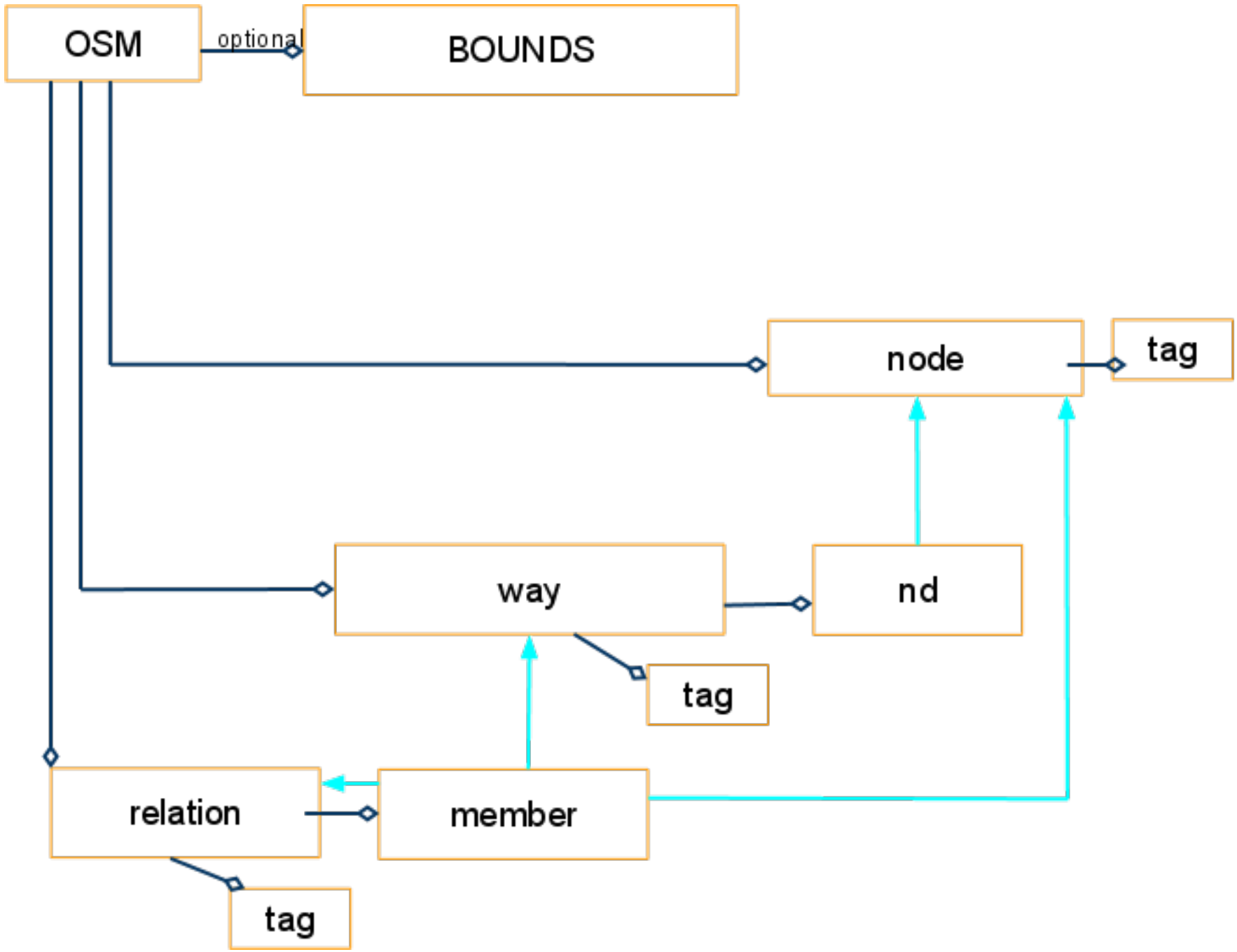
Component Overview

http://wiki.openstreetmap.org/wiki/Component_overview



source : http://wiki.openstreetmap.org/wiki/File:OSM_Components.png

OpenStreetMap XML format : .osm



OSM file format

The OSM xml file format is a simple, flat xml file that represents the graph of the world.

The root OSM node contains a bounds object the describes the area in the file. Then file then has a sequence of node, way and relation objects each with a set of tags and some sub objects. Ways reference nodes, and relations can reference ways, nodes or other relations.

See also :
http://wiki.openstreetmap.org/wiki/Data_Primitives and
http://wiki.openstreetmap.org/wiki/.osm

Tiles

Images are not considered dangerous to javascript.
You cannot read xml files from another server due to xss restrictions.
Images can be loaded from other servers with no problem. http://osmopenlayers.blogspot.com/2010/06/test-of-extrernal-svg.html

Tiles and tile rendering

There is an ongoing process of fetching map data via the API, and rendering maps as raster images known as tiles. Many new tiles need to be rendered to achieve full (global) coverage, but also updates to the underlying map data will mean that existing tiles need to be re-rendered.

Rendering

Marking a tile dirty **does not** mark all sub tiles as dirty. If you get 'More OpenStreetMap coming soon...' on a tile, it means there was no data for that tile and it is now in the queue to be rendered. You can find a tile status by getting a tile URL (right click and 'get URL for image' or similar):

- http://tile.openstreetmap.org/7/63/42.png

and then add /status on the end:

- http://tile.openstreetmap.org/7/63/42.png/status

which will tell you its creation timestamp and dirty status.
If you want to make a tile render before the seven day expiry then you can mark it as dirty by appending /dirty:

- http://tile.openstreetmap.org/7/63/42.png/dirty

Slippy Map

See Slippy Map. The main map appearing on the openstreetmap.org homepage is javascript interface letting you zoom and pan ("slippy" draggable panning). What happens is, the website (the 'index' view of the rails app) invokes OpenLayers. As it does so, it passes in a latitude and longitude based on the users last viewed location or URL params. OpenLayers does its client side javascript magic, to figure out which 'tile' images to fetch from the tile server.

Your own slippy map :

http://wiki.openstreetmap.org/wiki/Deploying_your_own_Slippy_Map

Export Embeddable HTML:
http://osmopenlayers.blogspot.com/2010/06/test-of-embedded-osm.html

open layers is the most common usage : OpenLayers (openlayers.org) is an feature-rich free open source JavaScript library.

Mapstraction (mapstraction.com) is an open source javascript wrapper around the above libraries (and several others) allowing you to swap your choice of map library provider without re-coding anything.

tiles@home

http://wiki.openstreetmap.org/wiki/Tiles@home

Osmarender - XSLTs which does OSM XML to SVG transformation.
tiles@home is a project which uses Osmarender to create a tile set for server a map layer alongside Mapnik's, but Osmarender can also be a good option for doing one-off renderings. This is the easiest way of getting maps in SVG format without any data dropped for better display, which allows you to do post-processing tidy up tweaks.

T@H has a server software, Tahngo (generation 2), running at the Tiles@home website, which get requests to render tiles from updated mapdata. There are many people who run the client software on their computers that ask what map-tile to render and contribute their results back to the server.
See also http://www.informationfreeway.org/

The world : Planet.osm

The world file, planet.osm, is an OSM XML File

http://planet.openstreetmap.org/

http://wiki.openstreetmap.org/wiki/Planet.osm

Osmosis can process the Planet.osm file to break it down into manageable chunks and to create/apply patches:
http://wiki.openstreetmap.org/wiki/Osmosis

Planet Mirrors

http://wiki.openstreetmap.org/wiki/Planet.osm#Mirrors

eg :
http://hypercube.telascience.org/planet/

extracts :
http://download.geofabrik.de/osm/
http://downloads.cloudmade.com/

An der Sandelmühle 35

http://www.openstreetmap.org/?lat=50.162994&lon=8.650918&zoom=18&layers=B000FTF

Example Node

<node id="34157406" lat="50.163021" lon="8.6511725" user="
MichaH" uid="8464" visible="true" version="3" changeset="
1841979" timestamp="2009-07-16T07:21:32Z">
<tag k="amenity" v="restaurant"/>
<tag k="name" v="Sandelmühle"/>
</node>

http://www.openstreetmap.org/browse/node/34157406

Example Way Building

<way id="30616450" user="MichaH" uid="8464" visible="true" version="2" changeset="
1841979" timestamp="2009-07-16T07:21:45Z">
<nd ref="338531695"/><nd ref="338531697"/><nd ref="338531700"/>
<nd ref="338531701"/><nd ref="338531703"/>
<nd ref="338531704"/><nd ref="338531705"/>
<nd ref="338531706"/><nd ref="338531702"/>
<nd ref="338531699"/><nd ref="338531698"/>
<nd ref="338531696"/><nd ref="338531695"/>
<tag k="addr:city" v="Frankfurt"/>
<tag k="addr:housenumber" v="35"/>
<tag k="addr:postcode" v="60439"/>
<tag k="addr:street" v="An der Sandelmühle"/>
<tag k="building" v="yes"/>
</way>
http://www.openstreetmap.org/browse/way/30616450

Example Relation : Urselbach Mühlenwanderweg

http://www.openstreetmap.org/browse/relation/78067

<relation id="78067" user="PHerison" uid="28378" visible="true" version="8" changeset="3304901" timestamp="2009-12-06T12:50:33Z">
<member type="node" ref="352783704" role=""/>
<member type="way" ref="4730851" role=""/>
.....
<member type="way" ref="23301887" role=""/>
<member type="way" ref="4913286" role=""/>
<tag k="name" v="Urselbach Mühlenwanderweg"/>
<tag k="network" v="lwn"/>
<tag k="operator" v="Stadt Oberursel"/>
<tag k="osmc:symbol" v="blue:white: :UM:yellow"/>
<tag k="route" v="foot"/>
<tag k="symbol" v="stilisiertes Mühlrad, Wasserlauf und Wasserflohkrebs"/>
<tag k="type" v="route"/>
</relation>

Example Way : An der Sandelmühle

http://www.openstreetmap.org/browse/way/4730848

http://www.openstreetmap.org/api/0.6/way/4730848

<way id="4730848" visible="true" timestamp="2010-05-30T07:57:35Z" version="9" changeset="4849228" user="baeuchle" uid="221387">
<nd ref="29536069"/><nd ref="30149135"/><nd ref="338531293"/><nd ref="30149137"/>
<nd ref="30149138"/><nd ref="338531300"/><nd ref="30149140"/><nd ref="440107927"/>
<nd ref="382390507"/><nd ref="440107939"/><nd ref="30149142"/>
<nd ref="30149143"/><nd ref="30149145"/><nd ref="30149146"/><nd ref="30149147"/>
<nd ref="30149148"/><nd ref="441278967"/><nd ref="30149149"/>
<nd ref="30149150"/><nd ref="30933456"/><nd ref="758595966"/>
<nd ref="43539957"/><nd ref="561211"/>
<tag k="highway" v="residential"/>
<tag k="name" v="An der Sandelmühle"/>
</way>

Osmosis

http://wiki.openstreetmap.org/wiki/Osmosis

Osmosis is the main tool for OSM processing.
It can read and write the osm file, read database, filter and split and merge areas. It is written in java and runs on the command line.

Importing data, extracting boxes, exporting data.

OSMosis extract box

http://wiki.openstreetmap.org/wiki/Osmosis#Extracting_bounding_boxes

```
bzcat downloaded.osm.bz2 | osmosis\
--read-xml enableDateParsing=no file=/dev/stdin\
--bounding-box top=49.5138 left=10.9351 bottom=49.3866 right=11.201 --write-xml file=-\
| bzip2 > extracted.osm.bz2
```

Potlatch

Potlatch is the online editor appearing on the 'edit' tab. Again the rails app has a view for the 'edit' tab. Potlatch is a flash object embedded on the web page. It is written in ActionScript. While running in the browser it makes calls to a special part of the API called the 'AMF Controller'. Because of browser security constraints it cannot be embedded on any website other than openstreetmap.org (since it must make HTTP requests to the API)

Mapnik

Mapnik is a rendering system which powers the display which is currently the Slippy Map default. The rendering process runs on the 'tile' server, and Mapnik tile images are served from that machine. This renderer takes its data from a postgres database (also on the tile server). This database holds data in the format expected by Mapnik, using PostGIS extensions. This is an entirely different format to the database used on the core OSM database server. The mapnik database is populated by occasionally running an osm2pgsql script on the weekly Planet.osm dump. Since Feb 2009 it is also being populated more regularly with the application of Osmosis diffs (hourly or even minutely)

Mapnik installation

Test installation :

Generate Tile :
python generate_tiles.py
bbox = (17.96,39.34,24.1,43.35)
render_tiles(bbox, mapfile, tile_dir, 1, 18, "World")

Hacked version of main page.
http://xhema.flossk.org:8080/mapdata/05/layerstest/OpenStreetMap.html

Using mapfish/openlayers
http://xhema.flossk.org:5000/customtiles.html

OSM API

OSM uses a REST API (i.e. OSM XML over HTTP, with use of simple URLs for object access, and standard HTTP response codes.) for interacting with the server.
http://wiki.openstreetmap.org/wiki/API_v0.6

It supports basic operations to read and write data to the server.

Before editing and data, users must authenticate against the server. All user changes are collected in a changeset, which is assigned a unique id.
The API logic is all part of the same Ruby on Rails application which powers the OSM front end website.

API

- **Read data :**
- **Get map data :GET /api/0.6/map?bbox=*left*,*bottom*,*right*,*top***
- get changeset GET /api/0.6/changeset/#id

- **Authenticated Data :**
- **Create Changeset: PUT /api/0.6/changeset/create**
- Close Changeset Close: PUT /api/0.6/changeset/#id/close
- create data : Create: PUT /api/0.6/[node|way|relation]/create
- Update data : PUT /api/0.6/[node|way|relation]/#id

Community

Mailing List
http://lists.openstreetmap.org/listinfo/talk

Forum:
http://forum.openstreetmap.org/

Wiki:wiki.openstreetmap.org

IRC Chat:irc.oftc.net #osm #osm-dev #osm-de

Routing

http://wiki.openstreetmap.org/wiki/Routing

Many routing services online : (OnlineRouters)

eg :http://www.openrouteservice.org

here is a bookmarked route:cloudmade routing

Other routing libs and other tools.

Free/Libre Open source software

The packages making up the core osm tools are under FLOSS licenses. There are many proprietary tools that use this data as well. Not all services around osm are free. Not all runtimes are free.
http://wiki.openstreetmap.org/wiki/WikiProject_FLOSS
Software used, packages available.

- POSTGIS
- Mapnik
- Josm / Merkaator
- GDAL/OGR
- Proj4
- Ruby / Perl / Python/ Javascript
- osm2pgsql
- shp2osm.pl

OpenStreetMap Licensing

OpenStreetMap creates and provides free geographic data such as street maps to anyone who wants them. The project was started because most maps you think of as free actually have legal or technical restrictions on their use, holding back people from using them in creative, productive, or unexpected ways.

What are the main differences between the old and the new license? The main reason for this is that maps can now be made with layers from incompatible data sources. Under the new license, they can put a map under any license they like, provided that they share any data enhancements they have made to our data

http://www.opendatacommons.org/licenses/odbl/

nominatim

http://wiki.openstreetmap.org/wiki/Nominatim

load the data again into a fresh postgis database with the gazeeter function and use a special php script to search

Nominatim Address Lookup

Nominatim indexes named (or numbered) features with the OSM data set and a subset of other unnamed features (pubs, hotels, churches, etc)

http://nominatim.openstreetmap.org/search?q=135+pilkington+avenue,+birmingham&format=xml&polygon=1&addressdetails=1

Nominatim install

http://osmopenlayers.blogspot.com/2010/06/setup-nominatim-for-postgis-84-on.html

1. Build software
2. Setup postgis
3. install php
4. load data with osm2pgsql --gazeteer
5. index the data
6. query it

PostGIS install

http://osmopenlayers.blogspot.com/2010/03/postgis-84-on-karmic-ubuntu-debian-gnu.html
and
http://osmopenlayers.blogspot.com/2010/04/installed-geos-322-and-postgis-15-from.html

Most of the instructions for postgis and postgres are out of date on the 8.4 / 1.5 versions.

Use a different port!:
/etc/postgresql/8.4/main/postgresql.conf
port = 5532

select the port :
export PGCLUSTER=8.4/localhost:5532

Nominatim Reverse Geocoding / Address lookup

Reverse geocoding generates an address from a latitude and longitude. The optional zoom parameter specifies the level of detail required in terms of something suitable for an openlayers zoom level.

http://nominatim.openstreetmap.org/reverse?format=xml&lat=52.5487429714954&lon=-1.81602098644987&zoom=18&addressdetails=1

geonames

http://www.geonames.org/

The GeoNames geographical database covers all countries and contains over eight million placenames that are available for download free of charge.

National Geospatial-Intelligence Agency's names

National Geospatial-Intelligence Agency's (NGA) and the U.S. Board on Geographic Names (most names except US and CA)

https://www1.nga.mil/ProductsServices/GeographicNames/Pages/default.aspx

Projections

http://docs.openlayers.org/library/spherical_mercator.html

Projections in GIS are commonly referred to by their “EPSG” codes, identifiers managed by the European Petroleum Survey Group. One common identifier is “EPSG:4326”, which describes maps where latitude and longitude are treated as X/Y values. Spherical Mercator has an official designation of EPSG:3785. However, before this was established, a large amount of software used the identifier EPSG:900913. This is an unofficial code, but is still the commonly used code in OpenLayers. Any time you see the string “EPSG:4326”, you can assume it describes latitude/longitude coordinates. Any time you see the string “EPSG:900913”, it will be describing coordinates in meters in x/y.

UTM Coordinates

http://en.wikipedia.org/wiki/Easting_and_northing
The terms **easting** and **northing** are geographic Cartesian coordinates for a point. Easting refers to the eastward-measured distance (or the *x*-coordinate), while northing refers to the northward-measured distance (or the *y*-coordinate). The orthogonal coordinate pair are commonly measured in metres from a horizontal datum.

Proj4 projection

CS2CS
http://proj.maptools.org/man_cs2cs.html

building proj with c++
http://github.com/h4ck3rm1k3/ProjC--

Example Projection

Examples :Lon :8.6511725 Lat: 50.163021 -->x:475085.2164415381 y:5556814.616540354 Zone :32N (epsg/32632)
http://home.hiwaay.net/~taylorc/toolbox/geography/geoutm.html
http://spatialreference.org/ref/epsg/32632/
+proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs to use , append +proj=tmerc -f "%.7f" cs2cs +proj=utm +zone=32 +ellps=WGS84 +datum=WGS84 +units=m +no_defs +proj=tmerc -f "%.7f"
475085.2164415381 5556814.6165403548
8.6511725 50.1630210 0.0000000

UTM Zones

http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system

The UTM system divides the surface of Earth between 80°S and 84°N latitude into 60 zones, each 6° of longitude in width and centered over a meridian of longitude. Zone 1 is bounded by longitude 180° to 174° W and is centered on the 177th West meridian. Zone numbering increases in an easterly direction.

import tools for GNS names

http://bazaar.launchpad.net/~kosova/+junk/openstreetmapkosova/annotate/head:/parsegeonames.pl simple import routine,
it does some basic parsing and creates an osm file based on the data. This was one of my first attempts at importing. Basic idea of transformation and creating negative ids for importing. the new nodes get unique negative ids, and then are loaded with josm into osm.

gdal and ogr

http://www.gdal.org/
is a translator library for raster geospatial data formats that is released under an X/MIT style Open Source license by the Open Source Geospatial Foundation. As a library, it presents a single abstract data model to the calling application for all supported formats. It also comes with a variety of useful commandline utilities for data translation and processing.

osm2blender 3d model

http://bazaar.launchpad.net/~kosova/%2Bjunk/openstreetmapkosova/annotate/head%3A/kmz_ImportWithMesh.py

http://permalink.gmane.org/gmane.comp.gis.openstreetmap.devel/19013

http://www.youtube.com/watch?v=Hw9aW3iI5fs

patches for osm2pgsql

http://github.com/h4ck3rm1k3/osm2pgsql

LibreDWG compiled with c++

http://github.com/h4ck3rm1k3/LibreDWGCPlusPlus

geocoder (usa) tool, autotoolized

http://github.com/h4ck3rm1k3/AutoToolsGeocoder/tree/master/geocoder/

autotoolize of mapnik and also reverse engineering

http://github.com/h4ck3rm1k3/MapNickAutotools

My first scripts for osm

http://bazaar.launchpad.net/~kosova/+junk/openstreetmapkosova/files

tons of my first tools to process openstreetmap data.

- forest walker to trace green data from landsat
- http://bazaar.launchpad.net/~kosova/+junk/openstreetmapkosova/files/head:/forestwalker/
-
- duplicate removal tools

Example kpaonline import

import tool for kpaonline data

How To | scraper

Extracting data from html, Projecting it to lat/lon processing it into osm.

1. use wget to get all the items into a directory :
each town is in kpaonline in a view from here:
http://www.kpaonline.org/AdminMun.asp?mun=Pristina
wget -r -l2 -dhttp://www.kpaonline.org/ http://www.kpaonline.org/AdminMun.asp?mun=Pristina

reverse engineering of the DirectDWG code

http://github.com/h4ck3rm1k3/InDirectDWG

patches for shp2osm

http://github.com/h4ck3rm1k3/shp2osm

twonickels with an dxf2osm tool

http://github.com/h4ck3rm1k3/TwoNickels

wikipedia scanner with openstreetmap point collector

https://code.launchpad.net/~jamesmikedupont/+junk/openstreetmap-wikipedia

advanced c++ template processing of openstreetmap data.

https://code.launchpad.net/~jamesmikedupont/+junk/EPANatReg

patch for inkscape to include openstreetmap photomapping feature

https://code.launchpad.net/~jamesmikedupont/inkscape/osm-inkscape

storage and splitting of osm files for git

http://gitorious.org/osmgit/osmgit-test

Torrent files for these:
http://osmopenlayers.blogspot.com/2010/06/first-running-of-osm-gittorrent.html

address database extractor for aspx database webpages

https://code.launchpad.net/~jamesmikedupont/aspxezxs/aspxtraktor

mapfish python

osm task

OSM Task:
We have a new tool to help mark the areas being worked on.
server is installed here, please check it out http://xhema.flossk.org:8881/
OSMTask is described here in brief.
http://wiki.crisiscommons.org/wiki/Camp_Roberts_Planning#OSM_Grid-Square_Checkout_Server_.28GCS.29_.28job_tasking.29

sahana

http://xhema.flossk.org:8000/sahana/default/index sahana

converting kml markers to osm

http://gitorious.org/macedonian_pois/macedonian_pois

my mapserver files

http://gitorious.org/albanianfloodingcrisiscamp

gpsbable to covert data

http://www.gpsbabel.org/
http://wiki.openstreetmap.org/wiki/GPSBabel

mapserver

http://mapserver.org

qgis

http://www.qgis.org/

map warper to overly photos

http://warper.geothings.net/

road matcher

http://wiki.openstreetmap.org/wiki/Roadmatcher

contour finding, cutting ways

http://osmopenlayers.blogspot.com/2010/03/first-version-of-way-cutter.html

gwt osm , google web toolkit osm

http://wiki.github.com/h4ck3rm1k3/GWTOsm/

http://wiki.openstreetmap.org/wiki/JOSM/GwtOsm
port of josm to gwt.
http://xhema.flossk.org:8180/GWTOSM/

http://osmopenlayers.blogspot.com/2010/06/gwtosm-example-output.html

STALLED porting josm to gjc

http://github.com/h4ck3rm1k3/josm

The porting of josm to gjc is an interesting project to help check the gnu compiler for java and the classpath.

Right now on hold for working on GWTOSM.

stalled : open address db

Controller/Geo/GeoNames.pm

here is a simple perl catalyst controller that does a geo names lookup and displays a openlayers map of the location found.
uses the GEONames Model
Model/Geo/GeoNames.pm

stalled :traindirector would be great if it could read osm files.

http://gitorious.org/osm-traindirector/debian-packages

I just started on the debian packaging, did not look into more details.

stalled : porting of mapzen code to haxe

http://gitorious.org/mapzen-port/mapzen-port

porting of mapzen code to haxe. this should be reworked to port to java