MapLocator

0.1

Software Design Specification

Prepared by

GS Lab

Great Software Laboratory Private Limited

Amar Arma Genesis,

8th Floor, Above Food Bazaar,

Baner Road, Baner,

Pune 411 045 INDIA

Tel: +91 20 4671 1000/10

Fax: +91 20 4671 1234

Table of Contents

1 Introduction

1.1 System Overview

GS Lab has developed "Map Locator" map-based web application platform that is generic and that can be used for a variety of purposes. The platform provides rich geographical visualization using the available Internet map servers like maps.google.com and maps.yahoo.com. It provides the ability to overlay custom geographical content on the base maps. And importantly, it has the ability for aggregating user content. Thus, any user will be able to locate an area and upload any geo-referenced location-based information to the Internet. User contributed content can then be aggregated and served on the Internet. The platform has a structure of roles that allows user content, content validation and content aggregation.

The platform has been developed as a multi-tiered architecture with

- PostgreSQL /PostGIS database to store GIS data,
- Custom PHP server along with UMN Mapserver & Apache 2.2 to extract the data from the database and process/ render data
- AJAX and OpenLayers based client that retrieves geometry/attribute data from PHP server, rendered image data from Mapserver to be visualized in the browser window.
- Drupal content management system to enable user participation and contribution.

The platform along with the application is deployed on the Ubuntu server.

1.2 References

The live site: *http://www.kmap.in*

2 Design Considerations

2.1 Assumptions

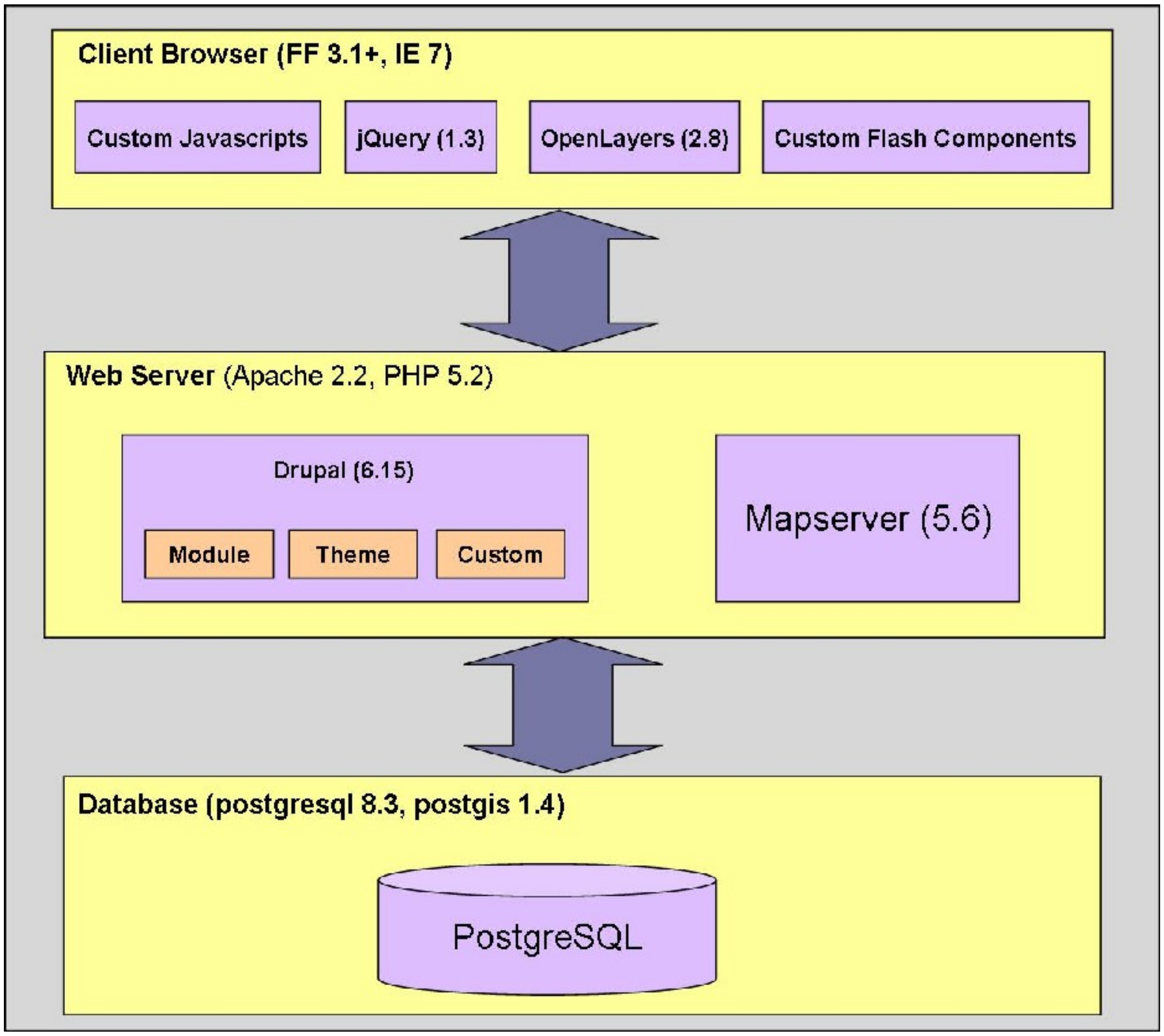
1. The end user browser supports and has JavaScript and flash 9.0 and above enabled.
2. The user has a high speed internet connection.

2.2 Constraints

1. The database must be an open-source freely available DBMS.
2. The code developed is aimed to be open sourced hence all technologies used development of the portal open-source freely available.

3 Architecture

The system follows multi-tier architecture.



3.1 Database layer:

The database layer stores all the data related to users and observations.

PostgreSQL is object-relational database management system, which is the back end of the system where all the data related to the portal is stored. I.e. layer details etc.

PostGIS provides predefined GIS data-types, APIs and tools for GIS specific data. In this system it is used to support geographical objects like topology etc.

3.2 Server:

The server is a Linux, Apache and PHP installation. The Apache server receives the requests from the client and passes them to PHP, which then processes it to generate the output.

Drupal:

Drupal is a free and open source content management system. It is mainly used for user management and access control in the system.

Theme:

A custom theme (mlocate) renders the UI. The theme is located inside appDir/sites/all/themes.

Module:

Various custom modules contain all the client code. The module is deployed in appDir/sites/all/modules.

Custom:

Server side php code is deployed in appDir/ location.

Mapserver:

MapServer is an open source platform for publishing spatial data and interactive mapping applications to the web. It uses WMS (Web Map Server) protocols to generate image of the requested layer. Whenever user selects a layer to view (raster or vector), the MapServer's job is to create an image and render it on the base map.

3.3 Client:

Browser:

Browser acts as an interface between the user and the system. Currently supported browsers are FF3.1 and above and IE7.

Custom JavaScript:

Custom Javascripts are used for map digitization (interface with OpenLayers), Ajax stubs (to send/fetch data from server) and UI rendering/modifications.

jQuery:

jQuery is a lightweight cross-browser JavaScript library that emphasizes interaction between JavaScript and HTML. It provides various plugins to access manipulate and animate HTML.

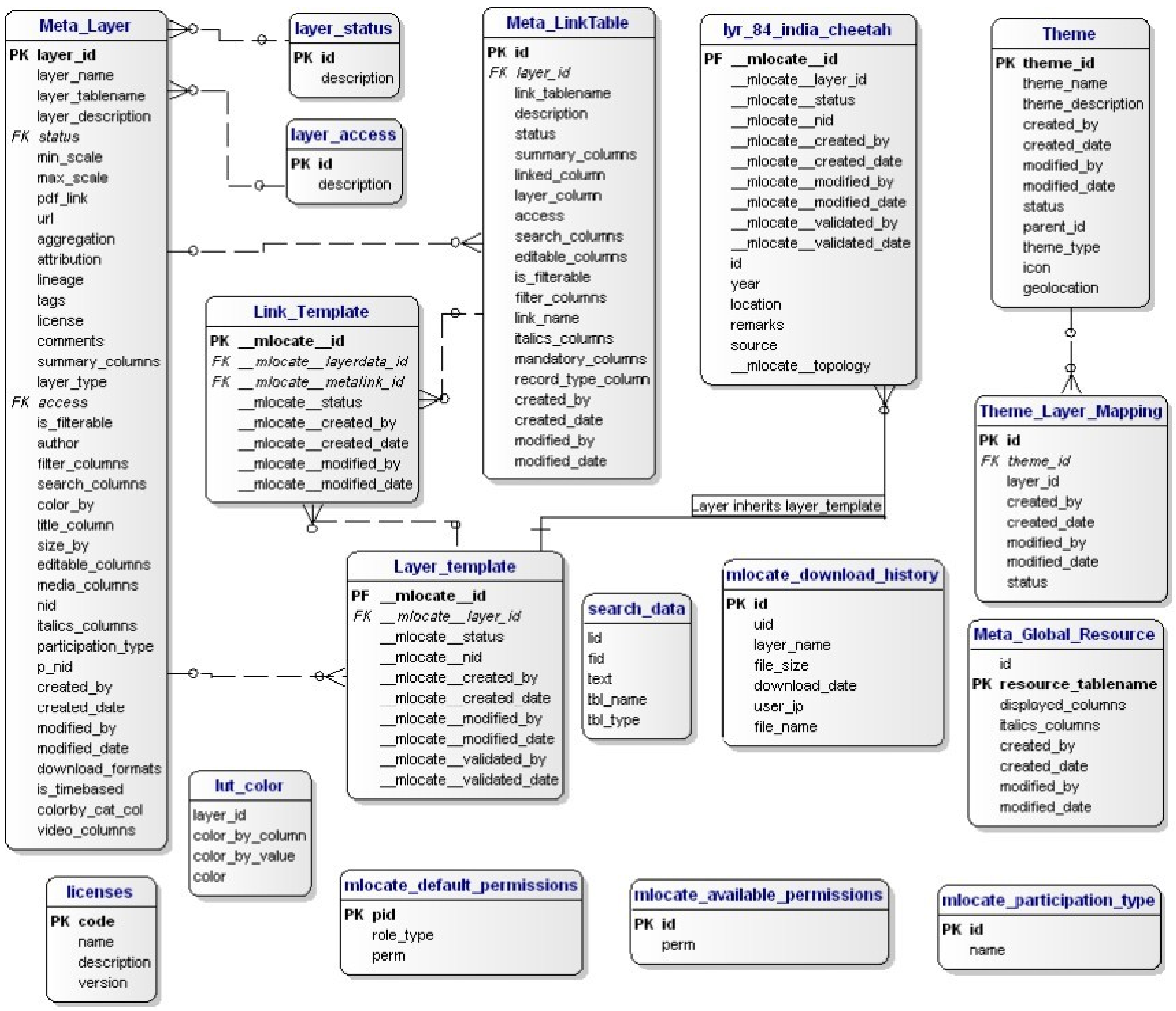
OpenLayers:

OpenLayers is a JavaScript library for creating GIS applications. It is a wrapper around currently available mapping services providing option to switch between providers. OpenLayers makes it easy to put a dynamic map in any web page. It is used for map digitization. I.e. showing layers, pop ups etc. It also acts as an interface for the MapServer.

4 Low Level Design

4.1 DB Design

MapLocator DB Schema Diagram:



The above diagram depicts the schema for the database for MapLocator. The tables displayed in above diagram are added on top of the basic Drupal tables in the DB.

MapLocator DB Schema Description:

A detailed explanation for individual table is as follows:

Meta_layer: This is the layer registry table. Whenever a new layer is added into the system an entry is made in this table. It stores information/attributes which apply to the entire layer.

Layer_template: This is an abstract/template table. As every layer in the system is stored as a separate table it has to inherit this table. This table has mandatory attributes/fields that every layer-table in the system should have. The diagram shows a layer table named "lyr_84_india_cheetah" which has been inherited from layer template.

Meta_LinkTable: This is the Link-table (explained below) registry table. Whenever a new link table is added into the system an entry is made in this table. Currently this table is not being used for MapLocator but can be used in future.

Link_Template: This is an abstract/template table. This table has mandatory attributes/fields that every link-table in the system should have. Link table is a table is used to store non-gis information, i.e. this table does not have a geometry column. A link table is linked to a layer table to store additional data a layer may have. Currently this table is not being used for MapLocator but can be used in future.

Theme: Theme table stores information about the theme. Theme table is used to categorize the layer's present in the system. There are two types under which layer/layers can be categorized – a) "by theme" and b) "by geography".

Theme_Layer_Mapping: This table stores the mapping between a layer and the theme. In simple words it stores the information – "which layer table belongs to which theme".

mlocate_download_history: This table stores information regarding download logs like user id, layer name, user ip address etc.

search_data: This table stores data which is referred by simple search feature. In other words whatever can be searched in "simple search" feature needs to be present in this table. The data in this table is seeded using a script which collects information from all layer tables and puts into this table. The columns to be used for search are mentioned in "search_columns" column in "Meta_Layer" table.

licenses: This table stores the list of license types (by-nc, by-nd etc) for records in the layer.

mlocate_default_permissions: This table stores access control information like roles (layer admin, layer validator etc) with respective permissions. The access control mechanism comes into picture with contribute feature enabled.

mlocate_available_permissions: This table stores access control information with respect to permissions. The access control mechanism comes into picture with contribute feature enabled.

mlocate_participation_type: This table stores a list of participation types a layer can have (public participatory, Restricted, No Participation etc).

Meta_Global_Resource, Global_Resource_Mapping: These tables are not being used by MapLocator. These tables are meant to be used as dictionary where a particular record in a layer/link table has reference/more information in the "Global Resource table". These tables work in a similar fashion as link table.

layer_access: This table stores information related to layer access (downloadable, non-downloadable). This is referenced in "Meta_Layer" in "access" column.

layer_status: This table stores information about status of the layer (active, inactive). This is referenced in "Meta_Layer" in "status" column. This parameter/value is switched off/on to make a layer visible/invisible on the portal.

4.2 Features

1. Base Map

A. Overview:

The base map provides the background visualization for the maps. These base map images are fetched from Google maps directly by the client browser based on the view port. You can choose from Google Physical, Google Satellite, Google Hybrid, and Google Streets, from the drop-down menu. On slow networks, fetching the Google images as tiles from the client could take some time. Apart from Google we have custom base map which fetched through Mapserver.

B. Implementation:

Files:

1. Client : map.js, includemap.js, maplayout.js
2. Server: Custom map file , page-map.tpl.php

Fetching base map is the first step in loading the map page of the portal. Base maps are defined as base layers in map object. This map object is created and initialized in a function “Intializemap ()” which is a starting point for client side processing. This function is defined in map.js file. In order fetch Google maps from Google server a map-key is required for authentication. This key is URL specific and needs to be generated by visiting maps.google.com. The key generated for “UAP” is used in the file “includemap.js” which includes the javascript to access Google maps. A custom base map “World Map” is displayed through Mapserver. Mapserver generates the map image through a configuration map file (.map extension) which is located in cgi-bin directory on the server.

The code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

2. Explore

A. Overview:

The portal consists of layers or maps organized by Themes and Geography. Themes show a collection of the maps into different groups that are relevant. Geography shows maps organized into an India level maps, and maps of specific locales. The layer tree on the left shows all the maps currently available on the portal. Click on a category to expand the tree and show a list of maps available in each category. Click on the check-box next to the map to display the map. The currently displayed map will be in bold and will show the top-most active map on the portal. The title of active map will also be displayed on the map above the map navigation toolbar. Features on this map are clickable. You can display multiple maps on the base map. All maps other than the active map displayed will be faded but will be visible on the map. Another tab”Participative Layers” shows a list of layers that are available for participation.

B. Implementation:

Files:

1. Client : map.js, maplayout.js
2. Server: category.php, page-map.tpl.php

The client side UI code is located in map.js while corresponding server side code is located in category.php.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element.

3. Search

A. Overview:

User can search for content that is available on the portal. The portal provides two categories of search:

Simple Search:

It is a simple string based search use to find the relevant content on the site which can be displayed on the map

Advanced Search:

There are following options under this category-

a) Geography Search (Location-based):
User can search for content around a location using following two options-

1) Bounding Box Search:

Users can search over particular location by marking an area on the map and search for the content inside the area.

2) Feature Based Search:

Users can search over particular location by selecting displayed feature/features (point, line and polygon) on the map and search for other overlapping features in other layers.

b) Attribute based Search:

Expert users can use this option to search using an attribute in a layer, condition and some value in the text box to create a conditional query. e.g. (list of locations where rainfall > 200 mm in the rainfall layer)

B. Implementation:

Files:

1. Client : MultilayerSearch(flash component),map.js, map/functions.js, uap/functions.js, maplayout.js
2. Server: MultiLayerSearch.php, page-map.tpl.php

MultiLayerSearch is a flash component which is loaded in a div. It communicates with the backend sever code existing in MultiLayerSearch.php to send search parameters and fetch the search result. This component also communicates with javascript code through "ExternalInterface.call ()" API and javascript communicates with this component through FAB (flex Ajax bridge), this required parameters/data is accessed to and fro from this component.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

4. Measurement Tool

A. Overview:

With this tool users can measure distance and area by marking points on the map.
Various measurement units supported are -
for distance: km, yards, miles
for area: km2, miles2, Acres, Hectares

B. Implementation:

Files:

1. map.js
2. maplayout.js
3. page-map.tpl.php

This feature is developed using OpenLayers functionality by using its various APIs and hence it is purely javascript based. The code to launch and control this feature exists in map.js.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

5. Contribute

A. Overview:

This feature allows a user to make a contribution by adding a point/line/polygon to an existing layer. A user must login to make any contribution, after logging in the UI shows a drop down which allows the user to select a layer from list of available layers (participative). Once user selects a layer a pop window with guidelines and action to be taken is shown.

B. Implementation:

Files:

1. Client: map.js, maplayout.js
2. Server: layerData.php, page-map.tpl.php

The client side UI code is located in map.js while corresponding server side code is located in layerData.php.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

6. Map Link

A. Overview:

Many a times user lands up in a situation where he/she has loaded some layers and have zoomed to a particular location on the map, now the user wants to share this exact map with his/her friends without requiring the friends to manually load the map, layers and zoom. By using this tool a user can simply copy-paste the url, which preserves the current state of the map, and share it with his/her friends

B. Implementation:

Files:

1. map.js
2. maplayout.js
3. page-map.tpl.php

map.js file includes the complete implementation of this feature

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

7. Layer Ordering

A. Overview:

The layer ordering window shows all the maps currently displayed. The top-most map in the window is the active layer. The active layer will also be shown in bold in the tree and is displayed on the map panel, above the map navigation icons. Note that only the top-most layer is clickable. The features of the active layer are also prominently displayed on the map, while all other layers are visible, but faded from view. The interface allows you to reorder the layers and change the active layer. To make any other map in the UI the active layer, click on the layer and drag it to the top of the list. Now this will be the active layer. It will be seen in bold in the layer tree, the name will be displayed on top of the map panel icons, the features in the map will be displayed brighter than the features of all other layers, and the features will be clickable. The layer ordering UI also has an information icon that will launch the map information pop-up. If layers are participatory, it shows a participatory icon. And you can close a layer and remove a map from the map view

B. Implementation:

Files:

1. Client: map.js, LayerOrdering(flash), maplayout.js
2. Server: page-map.tpl.php

Layer Ordering is a flash component which is loaded in a div. This component communicates with javascript code through “ExternalInterface.call ()” API and javascript communicates with this component through FAB (flex Ajax bridge), this required parameters/data is accessed to and fro from this component

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

8. More - Legend

A. Overview:

The legend tab is accessed by clicking on the tab in the right panel. This will bring out the legend tab. The legend is currently available only for polygon layers. The legend shows the categories and the color code for each category

B. Implementation:

Files:

1. map.js
2. maplayout.js
3. page-map.tpl.php

The code to display legend exists in map.js at the client. If the displayed layer is a polygon layer then the client sends a request to the Mapserver to fetch the legend, else if it is point layer then the name of the layer is displayed.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

9. More – Layer Data

A. Overview:

This feature displays a table with all the data for the features on the active map visible in the layer frame. The table allows sorting, searching, and setting pagination. The layer data will show a maximum of 250 features in the table. If there are more than 250 features of the active layer on the map frame, zoom in to see the data of a lesser number of features. The layer data table is linked to the layer. Thus clicking on a row in the layer data table will bring the pop-up of the feature on the map

B. Implementation:

Files:

1. Client : map.js, maplayout.js
2. Server: mapdata.php, page-map.tpl

The client sends a request to server to fetch information for the layer features currently displayed on the map. Client sends the bounding box i.e. the current view of map to the server and based on this information server fetches the features from the db and sends back to the client.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

10. More – Layer Info

A. Overview:

The layer information pop-up can also be accessed by clicking on the information icon near the layer name in the layer list or by clicking on the layer title on the map panel. The layer information pop-up lists all the data that is associated with the layer. It shows the attribution, the license if any, associated with the layer, the individuals who have worked on the data. Below the basic attribution and related data, it lists associated data with the layer. Click on the + to see the layer attributes. This panel shows the data that is associated directly with the layer. For each feature of the layer, there could be additional data. For example, at water body location there could be a list of bird sightings. These are organized as linked data to the layer. There could be one or more linked data. Click on the + to see details of linked data. For any column in the data, there could be some general information available. For example, if there is a species associated with the data, the species could have additional taxonomic, habitat, photo essay or any other information. This data does not concern the particular location, but is like a library resource on an item in the data. These are designated as resource tables. Click on the + to see the details available on the resource associated with map

B. Implementation:

Files:

1. Client : map.js, maplayout.js
2. Server: layerData.php, page-map.tp

The client sends a request to server to fetch information for the layer currently displayed on the map. The server fetches the information from the db and sends back to the client.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

11. More – View in Google Earth

A. Overview:

This feature enables a user to view the current map location in Google earth view i.e. 3 dimensional view. The code uses a Google earth plug-in provided by Mapfish.

B. Implementation:

Files:

1. Client : map.js, earth.js, maplayout.js, includemap.js
2. Server: page-map.tpl

The client (map.js) uses APIs from Mapfish (earth.js) to enable the Google earth plug-in. A URL specific key to access Google earth images needs to be generated from Google's website. Code to include the Google earth javascript along with key resides in includemap.js file.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element

12. More – Download

A. Overview:

This feature enables a user to download the active layer in various formats like kml, gml, shape and plain text. In order to download a layer the layer has to be downloadable and user has to login into the portal. The user has to select an option for download format and on click on download button fetches a tar file which contains layer data file along with the readme/metadata file.

B. Implementation:

Files:

1. Client : map.js, maplayout.js
2. Server: layerData.php, download.php, page-map.tpl

The client sends a request to server to fetch the layer to be downloaded. The server code (layerData.php, download.php) creates a tar of the layer along with metadata file and returns back to the client.

This is a pluggable feature and code to control the UI is located in page-map.tpl.php and maplayout.js. The UI visibility of this feature can be switched on by hiding/showing corresponding html element. Also event handlers can be added or removed to this element