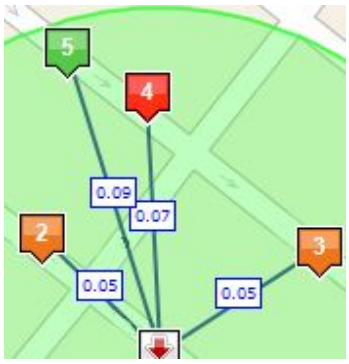


Netism Map Extreme 3.8.0
Google Maps and Bing Maps

User Guide and References

Netism Software

http://netismsoftware.com | info@netismsoftware.com

Table of Contents

1 Terminology

Latitude - Distance in degrees north or south of the equator. Ex: 40.68525.

Longitude - Distance in degrees east or west from Greenwich, England. Ex: -74.00331.

Geocode – A pair of latitude and longitude.

Geocoding - The process of assigning a geocode to a geographic feature.

Polyline – A set of straight lines connecting a set of geocodes, without forming a closed region.

Polygon – A set of straight lines connecting a set of geocodes, and forming a closed region.

Map Extreme (ME) – All license levels of the Netism Map Extreme module.

Module – An instance of the Netism Map Extreme module presented on a page.

Map – The Google or Bing Map control displayed from the Netism Map Extreme module.

Location – A record with/without address/geocode information, which is intended to be shown on a map. Ex: a portal, group, person, store, office, product, article, event and photo etc.

Icon/Pin/Pushpin/Balloon – An image or an icon displayed on a map to represent a location.

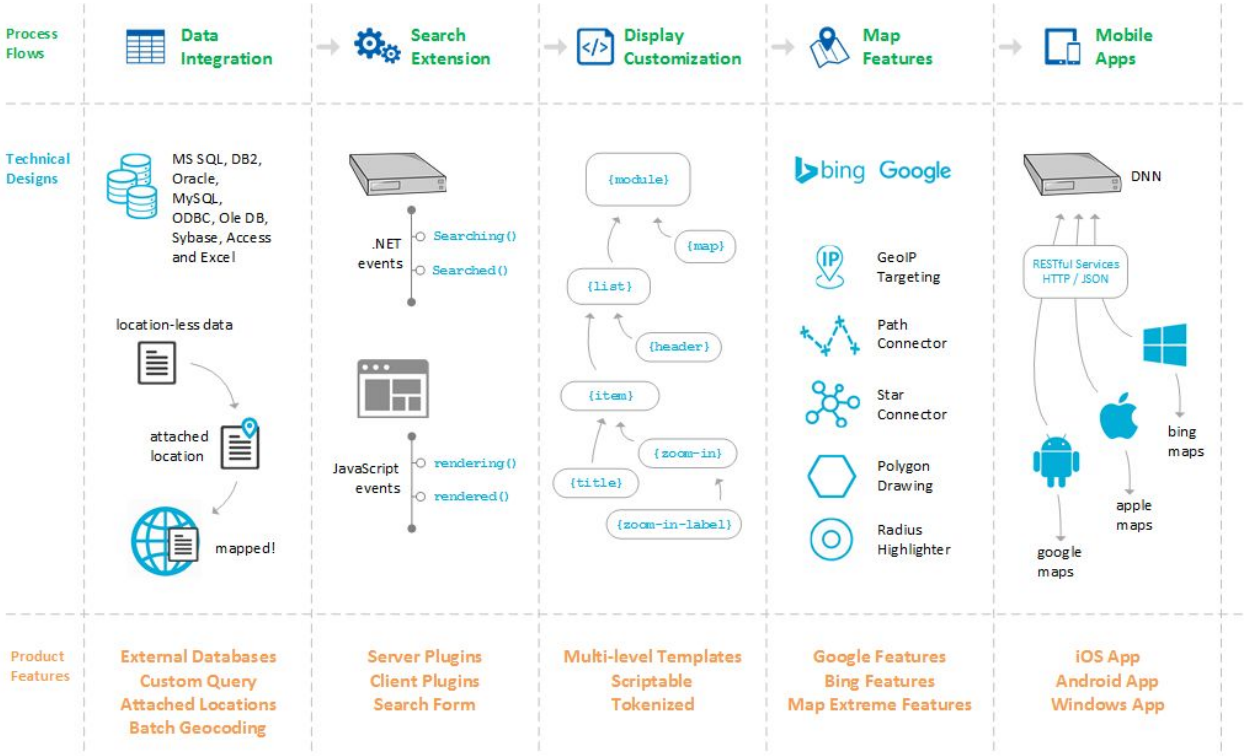
InfoBox/InfoWindow – The small window that pops up on mouse click or over an map icon.

2 Requirements

Map Extreme supports DNN 4+, IIS 6+, ASPX 2.0+ and SQL 2000+. It runs completely independent from Netism Software the company and the website itself. It's processed completely within your own server and the user web browsers. It uses the Google Maps or Bing Maps (formerly Virtual Earth), for rendering map tiles, navigation, directions, geocoding and pins display.

3 Product Overview

Map Extreme is the most powerful and flexible DNN map module. Advance features from Google and Bing. Total control over data, search and display. Native mobile apps. Easily formulate superior solutions for any mapping needs!



Mobile Apps

Map Extreme implements the DNN Web API framework and exposes data via HTTP RESTful services in JSON format. Native mobile apps are available to display the customized maps and data. The apps communicate with the Map Extreme instances from the DNN sites with the RESTful services.

In order to provide basic branding, we will package the apps with your logo, icon and DNN address. Apps are all natives (not HTML5) for best possible user experience. The set of apps

consists of an **iOS app**, an **Android app** and a **Windows app**. Additional fees may apply depending on app customizations, account provisions and store submissions etc.

Map Features

Map Extreme utilizes the differentiating features from both Google Maps and Bing Maps. Leveraging them as foundations, we also built a number of visually appealing and sophisticated map features. Such as Star/Path Connectors, Radius Highlighters, Animators, Visitor GeoIP Targeting, Location Autocomplete, Polygons drawing, Bing Collection, GeoRSS and KML etc.

Data Integrations

Custom queries offer unlimited possibilities while also simple to setup with the **Query Builder**. With external data providers, it reaches far beyond the local DNN database, to all Access, Excel, DB2, Sybase, ODBC, OLE DB, MYSQL, MSSQL and Oracle.

With the **Attached Location** and **Batch Geocoding** features, Map Extreme can attach locations to location-less data from any other DNN modules, so that they can be mapped as well.

Search Extensions

Map Extreme provides 30+ events. You can write plugins to inject custom logic to modify search behaviors and results throughout the entire process from server to browser.

The search form itself can also include any custom data fields and controls for search criterias.

Display Customizations

Map Extreme provides 50+ scriptable HTML templates processed by NVelocity. It is a sophisticated template engine with support of loops, conditions, variables and arithmetic. All display style and layout are totally customizable to match any look and feel.

4 Map Settings

Map setting form contains a set of configurable options that mainly specify how the map, locations and search should function and display. The map preview on map setting only provides a quick preview of the default map display region. It is not supposed to reflect the final map display.

Map Provider	Set the module to use Google Maps or Bing Maps.

Bing Key	The Bing Map Key, it is optional.
Latitude **Longitude** **Zoom**	If any of these fields is blank, then the map would automatically show the best view to display all the locations. For birdseye view, it's best to leave it blank. If "Use visitor location" is enabled, then it will override the settings here. The values for the 3 fields here will automatically be update as you drag around or zoom in/out the map.
Language	21 localized languages within the map display. Not all languages are supported in all countries. Even for the supported countries, the localized languages might not be available down to the city or street levels. If it's set to "DNN Page", then it will follow the DNN localization setting, or use the map default if the DNN localization does not match one of the below 21 localized languages. Czech – Czech Republic (cs-cz) Danish – Denmark (da-dk) German – Germany (de-de) English – Australia (en-au) English – Canada (en-ca) English – India (en-in) English – United Kingdom (en-gb) English – United States (en-us) Spanish – Mexico (es-mx) Spanish – Spain (es-es) Spanish – United States (es-us) Finnish –Finland (fi-fi) French – Canada (fr-ca) French – France (fr-fr) Italian – Italy (it-it) Japanese – Japan (ja-jp) Dutch – Netherlands (nl-nl) Norwegian – Norway (nb-no) Portuguese – Brazil (pt-br) Portuguese – Portugal (pt-pt) Swedish – Sweden (sv-se)
Unit	Set the map to use Miles or Kilometers.
Control	The size of the map navigation control: None, normal, small or tiny.

Style	The style of the map tiles: Road, Shaded, Aerial, Hybrid, Birdeye or BirdeyeHybrid.
Map Size Width/Height	The width and height of the map control display. Unit can be fixed (px) or dynamic (%). With dynamic %, the width and height are computed and set using the browser window dimensions each time the map is loaded.
Bing Map Collection CID	Locations, paths (polylines) and areas (polygons) saved from http://bing.com/map/. It's identified using a Collection ID. An example of CID is: "FA512DDc8164A00A!104"
GeoRss Url Import	URL to the GeoRSS feed. For Bing Maps, it must be on the same domain. The server must be configured to allow the file extension of the feed.
KML Url Import	URL to the KML feed. It must be publicly accessible.
Route Popup Window Properties	The popup window properties for the Routing Direction function.
List Page Size	The page size of the location list.
Use Custom Template	Set the module to use the custom templates.
Use Custom Query	Set the module to evaluate and execute custom queries. To debug, adding debug=query on the URL, which would display the executing query.
Use Visitor Location	Set the module to try to match the visitor location by the visitor IP address. It would then be used for the default map view and searching location. ● Download the Free GeoLite City database file: 1. Go to http://www.maxmind.com/app/geolitecity. 2. Click on the link "Download the latest GeoLite City Binary Format". 3. Extract GeoLiteCity.dat (27 MB) from GeoLiteCity.dat.gz (18 MB). 4. Upload GeoLiteCity.dat to the ME module folder. Example: C:\inetsrv\wwwroot\DesktopModules\NetismMapExt\GeoLiteCity.dat

	5. Uploading the file via FTP is recommended. To upload via the DNN FileManager, you need to allow .dat extension under host setting and watch out for page timeout error if you don't have a really fast upload speed.
	● Note that localhost or internal/private IP addresses would not work.
	● To simulate a visitor IP, add _ip=<ip address> on the URL. Try:
	  ● _ip=74.208.185.13 Wayne, PA
	  ● _ip=208.75.255.146 Fort Worth, TX
	  ● _ip=74.125.45.100 Mountain View, CA
	● To debug, add debug=geoip on the URL.
	● It offers over 99.5% accuracy on country level and 79% on city level for the US within 25 mile radius.
	More info: http://www.maxmind.com/app/geolite_city_accuracy

Reference jQuery	ME uses jQuery for certain functions. It's usually already referenced in DNN 5.x, but not in DNN 4.x. So unless jQuery is already referenced somehow, check "yes". If you are not sure, just leave it and check the map display, an alert message will be shown if jQuery is not referenced.
Enable Velocity	Velocity is a powerful template engine providing support of looping, conditioning, variables and arithmetic. Such sophisticated features require extra server resources, and should be used and turned on only when needed. It is off by default, and the tokens in the templates will just simply be replaced. More information on the design and syntax of the Velocity Template Engine: http://velocity.apache.org/engine/releases/velocity-1.5/user-guide.html
Auto-complete location bounding	When location auto-complete is turned on (insert reference), the location will be searched and suggestions will be displayed below the input field. By default, the suggestions do not favor any countries or regions. It will only show the suggestions from Google or Bing as is. Using this setting, you can limit the suggested locations to a region bounded by the TopLeft and BottomRight geocodes. You can enter the geocodes manually, or you can simply drag the map around, then click the "Copy to Auto-complete location bounding" link to populate the 4 fields. Note that as you drag the map around, the Latitude, Longitude and Zoom fields will be updated as well. Be sure to empty them before clicking the update button if you like them to stay empty.

Server Event Handler (Server Plugin)	The assembly name with a class that extends the ServerEventHandler base class.
Client Event Handler (Client Plugin)	The JavaScript URL, which contains any of the 15 event handling methods.
Star Connector	Set the map to draw a line between the searching location and all the matched locations, also optionally displays the distance on the line, forming a star.
Path Connector	Set the map to draw a line between all the matched locations, forming a path/polyline. It could optionally close and fill the path, forming a polygon.
Radius Highlighter	Set the map to draw a circle with the searching distance being the radius, showing the search boundary.
Location Animator	Set the map to automatically rotate and display the locations' infobox on the map based on a configurable second interval.
Custom field labels	On the add/edit location form, the module provides 5 custom fields, which you can use to store additional data. By default, their labels are "Field 1" and "Field 2" etc. You can provide custom labels by using the 5 custom field labels.
Location edit form script	Javascript code to be executed when the add/edit location form is loaded.
Map OnLoad script	Javascript code to be executed when the map and locations are loaded.

5 Map Locations

If the module is set to use custom queries, then this section does not apply. You can only manipulate locations via ME when the below 2 conditions are satisfied:

1. The module is not set to use custom queries.
2. The user is a portal admin, super (host) user or in the Edit Location Role.

Add location

Two ways to start:
1. Shift+ click on the map to add.
2. "Add Location" below the map, bottom of the container, or on the action menu.

Edit location

Click on the  icon to edit the location.

Delete location

Click on the Delete link below the edit location page.

Title	The title (or name) of the location, included in the pin popup.
Link	The URL to be linked for the title.
User Name	The name of the signed in user.
Display Order	The display order on the location list. Smallest number would make the location on the top, largest number would make it the last. Default is 0, it can be negative.
Active	Setting it "yes" would make it appear for public and under view mode of the page. Setting it "no" would only show it under edit mode of the page.
Address **City** **State/Region** **Zip/Postal** **Country**	Location information.
Latitude **Longitude** **Zoom**	Geocode of the location.
Plot Address on the Map	Filling the location information then click this would plot the location on the map and fill in the geocode automatically. The geocode and zoom level will also be automatically updated as you drag around or zoom in/out the map.
Icon	The URL of the custom icon image for the location to be used for the pin.

Field 1 - 5	Custom fields, up to 50 characters.
Description	Descriptions of the location. Unlimited characters length.

6 Data Integrations (Custom Queries)

You can use the Map Extreme add/edit location form, or define queries to select data from external data sources such as, Oracle, MySQL, ODBC, MS SQL and OleDB (Excel, Access, CSV and TSV etc). You can define multiple queries for a single module and match them by URL parameters and map zoom levels. Data returned from external sources can also be templated and searched. Bing Map Collection, GeoRSS Feed and KML documents can also be loaded on the map as external data sources.

You can easily attach location/geocode information to data from any DNN modules in the same database as the Map Extreme, such as:

DNN Core Objects:	Tabs, Portals, Users and Roles.
DNN Core Modules:	Events, Blogs, Repository, Forums, HTML, Document, Feedback, Announcement, and Forms and List (UDF) etc.
Commercial Modules:	Events Calendar, XMod, Business Directory, Dynamic Form, Active Social/Forum, Smarter-Thinker modules, CataLook Store, Property Agent, Engage Publish, DNN Articles, Media Gallery and Video Gallery etc.

There are 2 types of custom queries:

1) Map Locations – batch geocoding and map display.
2) Search Form – fields containing a list, such as drop down, checkbox list and radio list.

6.1 Custom Query – Map Locations

6.1.1 Map Location Query Properties

Match Order	The order in which the parameters of the query will be matched against the querystring parameters on the URL. If the same parameter is defined for more than one query, then the query of the first matched parameters will be executed.

Data Provider	DNN Default – The database the hosting DNN is using. MS SQL – Microsoft SQL Server. MySQL – MySQL Server. OLE DB – Excel, Access, CSV and TSV etc. ODBC – Older generations of data engine drivers. ORACLE – Oracle databases.
Connection String	The connection string to connect to the external data provider, blank for DNN default.
Parameters	URL querystring parameter names to match, multiple names are separated by commas. Ex: "country, state, city" or just "zip" etc. There are 2 special parameters: 1. Blank - default/fallback query. It is used when there's no match on all queries. 2. BatchGeocode – by entering "BatchGeocode" for the match parameter on a query, it will be used to pull address information for the batch geocoding form.
Zoom Levels	Map Extreme will pass the zoom level of the map control to the server to match a query in addition to the parameters. This support allows the module to execute different queries to retrieve and display different data when the map is on different levels (such as on the countries levels, versus, on the street levels etc.) If it's not necessary, then just leave it as 1 to 19.
Query	The query to retrieve location data. The order by expression should not be included here.
Order By	The order by expression of the query. Do not include table alias, just the column names and order direction like: "age desc, name" instead of "emp1.age desc, emp2.name"

Custom query can be used with both the default or custom templates. The query results will be loaded in a **System.Data.DataTable** object. All DataColumn can be referenced as tokens in the templates. If you have a query: "`select count(*) as total, city …`", then you can use {total} in custom templates. All token names not found in the DataTable will be ignored. Similarily, all DataColumns not referenced in tokens will also be ignored.

Custom Query With Default Templates

The default templates expect the query to return the below fields,

like from the Netism_MapExtreme_Locations table:
LocationID, Title, Link, Description, Address, City, State, Zip, Country, Latitude, Longitude.

Custom Query With Custom Templates

Custom query must return at least *Latitude, Longitude and LocationID* for proper map display.

6.1.2 Map Location Query Execution

Map Extreme performs a series of processing to select, inspect, transform, enrich and execute the map location query. Below table explains the order and details of the entire process.

Make sure to set "Use Custom Query" to "Yes" under Map Setting.

Process	Note / Sample
1) Query Selection	Match Query by Parameters and Zoom Level. If all the queries with parameters don't match the current URL, then the query with no parameters is used. See "Parameters" in 6.1.1.
2) { and } escaping	If { or } is intended to be a literal character, you can escape { by {{, and } by }}.
3) Tokens Replacement	{PortalId}, {TabId}, {UserId}, {ModuleId} and {url-param-name}.
4) Template Processing	If the query contains any Velocity template language, it is processed at this stage.
5) Security Inspection	For security reasons, queries with comments or any of the below list of keywords would be denied: d*rop, truncate, delete, insert, update, set, exec, execute, create, alter*
6) Provider Selection	Based on the Data Provider field of the custom query.
7) Query Enrichment	Enriching the query with additional logic for distance calculation, filtering, paging, counting total and sorting.

You can put **debug=query** on the URL to see the final executing query displayed in place of the location results. It will be updated as you change the criterias on the search form too.

6.1.3 Map Location Query Sample

Below table illustrate the process of a sample query.

Parameters: country
Query: `Select * from tbl_WorldCities where country = '{country}'`
Order By: city

Process	**Note / Sample**
1) Query Selection	URL: /tabid/123/default.aspx?country=USA Matched Query: `Select * from tbl_WorldCities where country = '{country}'`
2) Security Inspection	OK
3) { and } escaping	OK
4) Tokens Replacement	Base Query: `Select * from tbl_WorldCities where country = 'USA'`
5) Provider Selection	`Default DNN Data Provider`

6) Query Enrichment	`… … … ;WITH NetismMapExtremeLocations AS`
	`(`
	`SELECT *,`
	`distance = @EarthRadius * … … …`
	`FROM (`
	`select * from tbl_WorldCities where country = 'USA'`
	`) AS nme_query`
	`),`
	`Searched AS`
	`(`
	`SELECT row_number() OVER ( ORDER BY` **`city`** `) … … …`
	`FROM NetismMapExtremeLocations`

	WHERE distance <= **5 AND state = 'NY'**) … … … RowNumber BETWEEN **1** AND **5**

6.2 Custom Query – Search Form

6.2.1 Search Form Query Properties

Data Provider	DNN Default – The database the hosting DNN is using. MS SQL – Microsoft SQL Server. MySQL – MySQL Server. OLE DB – Excel, Access, CSV and TSV etc. ODBC – Older generations of data engine drivers. ORACLE – Oracle databases.
Connection String	The connection string to connect to the external data provider, blank for DNN default.
Name	The name of the query, the search form field will reference this query by the name.
Query	The query to retrieve search field items. The first column is used for the value of the item; the second column is used for the text of the item. If there's just one column, then it's used for both.

6.2.2 Search Form Query Execution

Map Extreme performs a series of processing to select, inspect, transform and execute the search form query. Below table explains the order and details of the entire process.

Process	**Note / Sample**
1) Query Selection	Match Query by the Name.
2) { and } escaping	If { or } is intended to be a literal character, you can escape { by {{, and } by }}.

3) Tokens Replacement	{PortalId}, {TabId}, {UserId}, {ModuleId} and {url-param-name}.
4) Template Processing	If the query contains any Velocity template language, it is processed at this stage.
5) Security Inspection	For security reasons, queries with comments or any of the below list of keywords would be denied: d*rop, truncate, delete, insert, update, set, exec, execute, create, alter*
6) Provider Selection	Based on the Data Provider field of the custom query.

6.4 Query Builder

Many types of records can be associated with a geographical location, such as a video, a blog, a news article, an event and a user etc. Many modules handle them very well. However, most of them don't offer a function to even just simply display them on a map, let alone support of templating the map view and searching by distance etc.

Query Builder offers a quick and easy way to create a query that retrieves data from certain modules for certain portals or module instances, and attach location and geocode fields to them. It is mainly used to build queries for the Batch Geocoding form. It only works on the same DNN database that is running SQL 2005 or up.

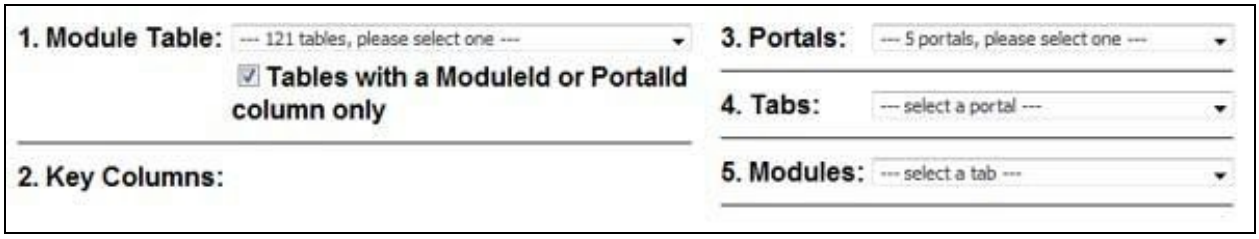
It selects data from a table that is used by another module, and then left joining them to one of the Map Extreme tables on the table name and the primary keys. The table Netism_MapExtreme_AttachedLocations would then contains address and geocode fields for the records from the other modules.

Once a query is defined, you can use it on the Batch Geocoding form to enter the address, and it would geocode all of them with a single click. You will then be able to use the same query to retrieve records and geocodes to accurately display them on a map.

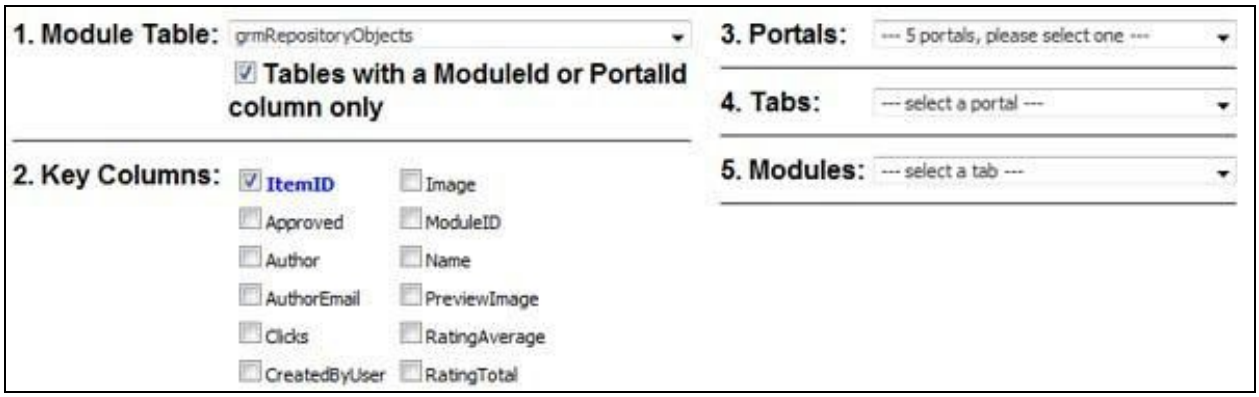
6.4.1 Query Builder Sample

Attach location information to a Repository Module

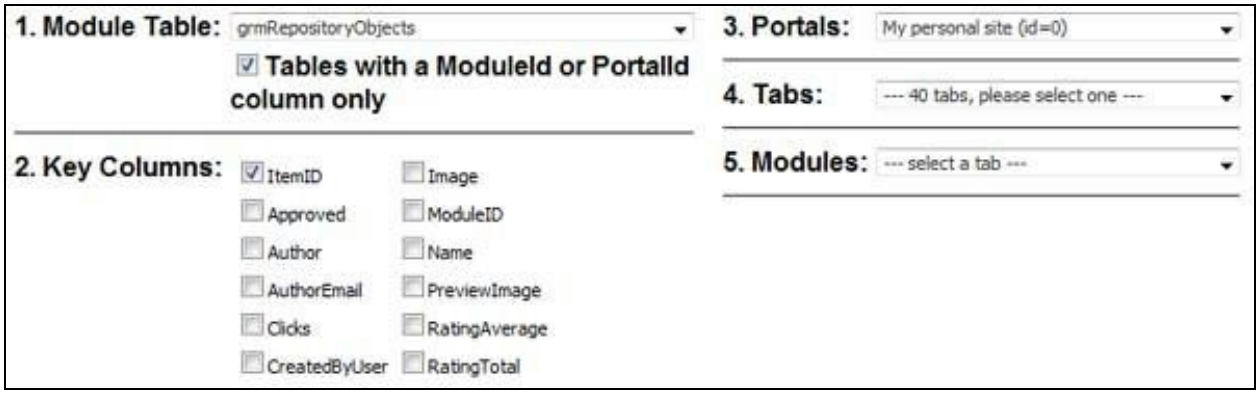
Start by clicking on the "Start Wizard!" button. The wizard loads all tables from the DNN database. In this case, the Repository Items, which is the grmRepositoryObjects table.



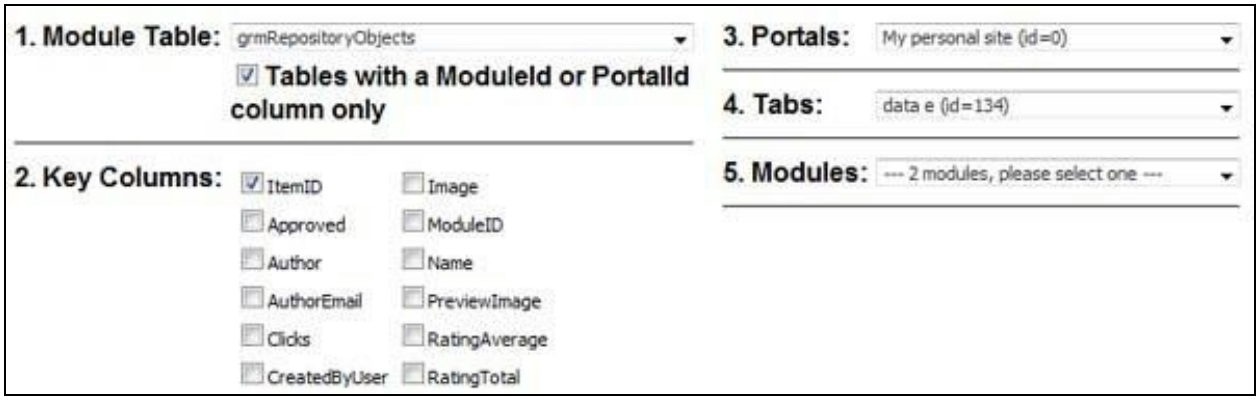
Selecting the table would load all columns with the Primary Key and Identity columns selected for the Key Columns. The table name (grmRepositoryObjects) and the key columns (ItemID) would be used for joining condition to join the Netism_MapExtreme_AttachedLocations table.



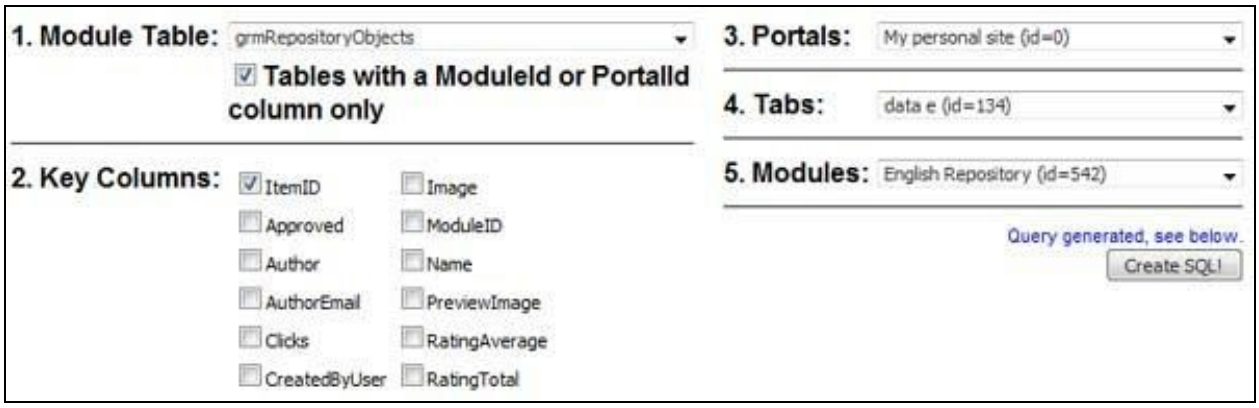
Select a portal from the Portals dropdown, in this case “My personal site.”
All tabs on the selected portal will be loaded.



Select a tab (or page) from the Tabs dropdown, in this case “data e.”
All modules on the selected tab will be loaded.



Select a module from the Modules dropdown, in this case "English Repository." The query will be generated, selecting only the repository items for the module instance with the ID of 542. All records are attached with the address and geocode fields matched with their ItemID.



6.4.2 Query Sample – Repository Module

The generated query by the wizard:

```
SELECT t.*,
       ISNULL(m.TableName, 'grmRepositoryObjects') AS TableName,
       ISNULL(m.ID1, t.ItemID) AS ID1,
       m.Address, m.City, m.State, m.Zip, m.Country,
       m.Latitude, m.Longitude, m.Zoom, m.Iconfile
FROM   grmRepositoryObjects AS t
LEFT JOIN
       dbo.Netism_MapExtreme_AttachedLocations AS m
       ON m.TableName = 'grmRepositoryObjects'
          AND t.ItemID = m.ID1
WHERE  t.ModuleID = 542
```

You can now save it as a Map Locations query, enter BatchGeocode for the Parameters. Create another Map Locations query, enter the same query and leave the Parameters blank. With the

2 queries setup, you can use one to fill in location information for the repository items then batch geocode them on the BatchGeocoding page. And use another query to pull the items, supposedly, already geocoded, for the map display.

Optionally, you can add "`and isnull(m.Latitude,0)=0 and isnull(m.Longitude,0)=0`" for the BatchGeocoding query, so only records not geocoded would show up for geocoding. And you can add "`and isnull(m.Latitude,0)<>0 and isnull(m.Longitude,0)<>0`" for the default query, so only records already geocoded would show up for the map view.

Furthermore, you can add "`SELECT TOP 100 t.*, …`" instead of "`SELECT t.*, …`", so you can geocode only 100 records at a time if you have a lot.

However, some modules, due to their flexibilities in design, don't have a database table structure that our Query Builder could support and properly build the queries with the intended fields. Some examples of those modules: Active Social, XMod, Property Agent, Forms and Lists (or User Defined Table) and DNN User Profile etc. Or sometimes, more logic is required for additional computation or aggregration. In those cases, you can either write the custom queries yourself, or you can always contact us.

If you write the query, it must return the TableName and ID1 columns as generated by the Query Builder. It optionally support up to three Key columns with ID2 and ID3. ID1, ID2 and ID3 can be string.

6.4.3 Query Sample – Computation/Aggregation

Below is a sample query to first group events by their locations, and then use the location as ID1 to join it to our table attaching the location information. This way, not only you can display individual records, you can also display computed, aggregated or statistical information like the total field from the below query. Ex: "31 events coming up in Soho, New York", "162 photos in this album" and "31 people in this group" etc.

```
SELECT t.*,
       ISNULL(m.TableName, 'Events') AS TableName,
       ISNULL(m.ID1, t.location) AS ID1,
       m.Address, m.City, m.State, m.Zip, m.Country,
       m.Latitude, m.Longitude, m.Zoom, m.Iconfile
FROM
(
      select location, count(*) as total from Events group by location
) as t
LEFT JOIN
       dbo.Netism_MapExtreme_AttachedLocations AS m
       ON m.TableName = 'Events'
          AND t.location = m.ID1
```

6.4.4 Query Sample – Name/Value Pair Structure

A query to select the DNN users and their locations for BatchGeocoding. Similar querying techniques are required for modules using name/value pair database structure.

```
declare @portalid int
select @portalid = 0

select ISNULL(m.TableName, 'Users') AS TableName,
       ISNULL(m.ID1, u.UserID) AS ID1,
       u.userid as LocationID, u.displayname as Title,
       '' as description, '' as link,
       m.Latitude, m.Longitude, m.Zoom, m.Iconfile,
       up1.PropertyValue as Address,
       up2.PropertyValue as City,
       up3.PropertyValue as State,
       up4.PropertyValue as Zip,
       up5.PropertyValue as Country
from  Users u
join  UserPortals p on u.UserID = p.UserID and p.PortalID = @portalid
left  join UserProfile up1
  on  up1.UserID = u.UserID
  and up1.PropertyDefinitionID =
dbo.GetProfilePropertyDefinitionID(@portalid, 'Street')
left join UserProfile up2
  on  up2.UserID = u.UserID
  and up2.PropertyDefinitionID =
dbo.GetProfilePropertyDefinitionID(@portalid, 'City')
left join UserProfile up3
  on  up3.UserID = u.UserID
  and up3.PropertyDefinitionID =
dbo.GetProfilePropertyDefinitionID(@portalid, 'Region')
left join UserProfile up4
  on  up4.UserID = u.UserID
  and up4.PropertyDefinitionID =
dbo.GetProfilePropertyDefinitionID(@portalid, 'PostalCode')
left join UserProfile up5
  on  up5.UserID = u.UserID
  and up5.PropertyDefinitionID =
dbo.GetProfilePropertyDefinitionID(@portalid, 'Country')
left join Netism_MapExtreme_AttachedLocations AS m
  on m.TableName = 'Users'
  and u.UserID = m.ID1
```

7 Application Extensions (Plugins)

Map Extreme defines 30 interface methods. Custom .NET/JavaScript code can be written to implement those methods to alter almost all data and behavior.

7.1 Server Event Handlers/Plugin

To create a server-side plugin:

1. Create a VS.net project
2. Reference DLLs: DotNetNuke.dll, NVelocity.dll and Netism.MapExtreme.dll.
3. Import below namespaces.
    a. System.Web
    b. System.Data
    c. System.Collections.Generic
    d. DotNetNuke.Entities.Modules
    e. NetismSolutions.DnnModules.NetismMapExtreme.Entity
    f. NetismSolutions.DnnModules.NetismMapExtreme.Util
    g. NVelocity
    h. NVelocity.App
4. Inherit from the class ServerEventHandler.
5. Override and implement any of the below 15 methods, be sure to include the "override" keyword in the method declaration.
6. Compile the project, place the DLL in the /bin folder.
7. Enter the DLL name under Map Setting.

To troubleshoot, include the string "&debug=handler" on the URL. The module will display the name of the DLL it's configured to load and execute. The export process of map extreme will include the name and the binary of the configured DLL in the exported XML file. The import process will create the DLL and place it in the bin folder if the file does not already exist there.

The ServerEventHandler provides the 2 properties below:

Portal	an instance of DotNetNuke.Entities.Modules.PortalModuleBase, which references the current executing module.
Context	an instance of System.Web.HttpContext, which hold context information like the Request and Response object of the current page execution.

```
void MapInfoLoaded(MapInfo MapInfo)
```

This method is called once after the MapInfo object is created early in the process. MapInfo contains all the settings defined on the Map Settings page. Any changes to the object will be carried thoughtout the process and to the JavaScript/Ajax layer.

```
void StandardTokensLoaded(Dictionary<string, string> Tokens)
```

This method is called after the standard tokens are loaded. Standard tokens are the TabID, PortalID, UserID, ModuleID and the querystring parameters. They are available to all templates, and are refreshed many times in the process. As such, this method is also called many times. You can use this method to alter the standard token values, or add your own standard tokens.

```
void TemplateLoaded(Dictionary<string, string> Templates)
```

This method is called once after the default/custom templates are loaded from the database/files, and before they are parsed and processed.

```
QueryInfo QueryMatching()
```

This method is called once before the module tries to match a default or custom query to execute. You can override the logic by returning your own QueryInfo object. Below table shows how to set the properties of the object.

QueryInfo Class	
Provider	See “Data Provider” in 6.1.1. Leave this empty for data from the same DNN database, or one of below string for external data: mssql, mysql, oledb, odbc, oracle.
ConnectString	See “Connection String” in 6.1.1 Required for external data sources.
Query	See “Query” in 6.1.1
OrderExp	See “Order By” in 6.1.1

```
void QueryMatched(QueryInfo QueryInfo)
```

This method is called once after a custom query is matched for execution when the module is set to use custom queries. The query is now parsed, inspected, token replaced, template processed, and ready for execution. However, this is still not the

final executing query, see 6.1.2. The QueryInfo object could be null if there's no match, or the matched query failed security inspection.

You can use this method to transform or inject additional logic to the query. For example, you can use the JavaScript NetismMap_Searching() (see 7.2) to append additional querystring parameters to the searching URL, and then inspect the parameters here to add more logic to the query. Below is a simple example:

```
If ( Context.Request.QueryString["my_flag"] == "1" )
   QueyInfo.Query = QueyInfo.Query.Replace("/* dynamic-logic */",
                                           "and type = 1");
```

The place holder is included as SQL comments, so if they are not replaced, they will just get ignored. It provides a convinence way to inject dynamic logic. In above example, if the original query is:

```
select * from tbl_locations where state = 'ny' /* dynamic-logic */
```

It will becomes:

```
select * from tbl_locations where state = 'ny' and type = 1
```

```
DataTable LocationSelecting(QueryInfo QueryInfo)
```

This method is called once the executing query is generated, and before executing it to create the DataTable. By overriding this method and returning a DataTable, the module will bypass the default DataTable loading logic, and will use the returned DataTable for all subsequent processing.

```
void LocationSelected(QueryInfo QueryInfo, DataTable Locations)
```

This method is called after the query is executed and a DataTable is loaded. It can be overrided to enrich/expand the data in the DataTable.

```
int TotalCounting(QueryInfo QueryInfo)
```

If LocationSelecting() return a custom DataTable, and if it holds only a sub-set of paged results, then this method can be overrided to return the total count based on the custom logic.

```
void HandleAjaxEvent(HttpContext ctx)
```

This method is called when the below page is called. Request and Response objects can be access through the ctx parameter. This method has the complete control of the page processing and rendering.

7.2 Client Event Handlers/Plugin

```
NetismMap_MapScriptLoaded()
```

```
NetismMap_MapControlLoaded(setting, map)
```

```
NetismMap_ModuleLoaded(setting, map)
```

```
NetismMap_Searching(setting, map, url)
```

```
NetismMap_ResultDisplaying(setting, map, html)
```

```
NetismMap_ResultDisplayed(setting, map, search, result, pager1, pager2)
```

```
NetismMap_MenuBinded(setting, map)
```

```
NetismMap_MenuSelecting(setting, map, select)
```

```
NetismMap_MenuSelected(setting, map, select)
```

```
NetismMap_MenuClearing(setting, map, select)
```

```
NetismMap_MenuCleared(setting, map, select)

NetismMap_LocationGeocoding(setting, map, location)

NetismMap_LocationGeocoded(setting, map, results)

NetismMap_MapObjectAdding(setting, map, type, obj)

NetismMap_MapObjectAdded(setting, map, type, obj)

NetismMapAjaxGet(params, callback)
```

8 Display Customizations (Custom Templates)

With custom templates, except for the map tile images, you can virtually change all layout and style display you can see with your own layout and styles. The template engine recognizes some special instructions in the templates text which are called “tokens.” It will replace the recognized tokens with values that could be specific and unique to each map view and search. Tokens are enclosed with the { and } characters. If you need to display the { or } character, then use {{ for { and }} for }. Unrecognized tokens would be removed.

Make sure to set “Use Custom Template” to “Yes” under Map Setting.

8.1 Overall Module View

8.2 Map Display Area

8.3 Location List Item

8.4 Location List Pager

8.5 Map Icons/Infobox

8.6 Polygon Control Panel

8.7 Routing Direction

8.8 Connectors / Highlighters

8.9 Export / Print

8.10 XML Download

8.11 Search Form

See section 9.

There are 3 main types of tokens:

Map Tokens

Or standard tokens, unique only for each map/search view.

{Total}	Total number of locations returned.
{PortalId}	ID of the current portal.
{TabId}	ID of the current tab (or page.)
{UserId}	ID of the logged in user, or empty if not available.
{ModuleId}	ID of the module.

URL Parameters – All parameters on the URL Querystring can also be referenced.
So for the URL like: /map.aspx?zip=12345&type=dealers&speak=Spanish&rate=5
You can reference the parameters by {zip}, {type}, {speak} and {rate}.

Field Tokens

Or location tokens, unique for each map locations. The values would be retrieved from the database unchanged. There are 2 types of field tokens:

Map Extreme Default Query – fields from the Netism_MapExtreme_Locations table.

{LocationID}, {Title}, {Link}, {Description}, {Address}, {City}, {State}, {Zip}, {Country}, {Latitude}, {Longitude}, {Zoom}, {Iconfile}, {Username}, {Ext1}, {Ext2}, {Ext3}, {Ext4}, {Ext5}, {SortOrder}, {IsActive}, {CreatedDate}

Custom Queries – fields from a query defined in the Custom Queries page.

All fields returned could be referenced by tokens, the fields can be physical fields directly from a table column, or dynamically aggregated, computed or aliased fields.

So for the query: “select categoryid, count(*) as articles, avg(rating) as average ...”
You can reference the fields by {categoryid}, {articles} and {average} respectively.

Processed Tokens

They can be unique to each map view or locations. They are results of either combining other tokens result or mixing with some HTML/Javascript code. Note that not all processed tokens are available for all templates.

Module view tokens

{map} Display code for the map view. It processes values and results from the Map Setting, Permissions and Icon templates.
{list} Display code for the location list. It combines all list templates results.
{pager} Display code for the pager bar. It combines all pager templates results.
{search} Display code for the search form. Code results from the Search Form.

Pager tokens

{last-link} JS function call to activate the last page for the location item list.
{next-link} JS function call to activate the next page for the location item list.
{last}, {next} Display code for the last and next links, it uses {last-link} and {next-link}. You can change to use your own text or image in the templates.
{from}, {to} The index of the locations on the current page.

Location tokens

{Index} Index of the location.
{link-directions} Display code for the direction link.
{link-edit} Display code for the location edit link.

{zoom-satellite-js} JS function call to activate the satellite view for the location.
{zoom-street-js} JS function call to activate the street level zoom.
{zoom-city-js} JS function call to activate the city level zoom.
{zoom-region-js} JS function call to activate the region level zoom.

By using the above 4 tokens, you can use your own display code to trigger the functions.

{zoom-satellite} Display code for the satellite view link.
{zoom-street} Display code for the street level zoom link.
{zoom-city} Display code for the city level zoom link.
{zoom-region} Display code for the region level zoom link.
{icon-idle} Icon image URL for the location.
{icon-active} Icon image URL for the location for when it's selected.
{popup-show} JS function call to show the location popup inside the map.
{popup-hide} JS function call to hide the location popup inside the map.
{distance} Place holder for the calculated distance from a location search.

There are 4 main types of templates:

Module View Template

The overall layout of the module is customizable by placing the tokens in different places. If the search form is not needed, just take out {search}. If the pager is not needed, just take out {pager}. If you like the search form to be on the left, on the top, or the list to be on the top or bottom, just change the HTML table structure accordingly.

Available tokens

All map tokens, {map}, {list}, {search} and {pager}

Default template code

```
<table><tr>
<td valign="top">{map}</td>
<td valign="top">{search}<hr/>{list}<hr/>{pager}</td>
</tr></table>
```

Pager Templates

Result of all pager templates is referenced by {pager} in the module view template. If a search returns no matches, {pager} will use the No Match template. If there are results, it will use the Pager template. On any given page, if there's a last page available, {last} would use the Last Active template, otherwise it would use the Last not Active templates, same idea for {next}.

Pager	`{last} | {from} - {to} of {total} | {next}`
Last Active	`<a href="#" onclick="{last-link}">last</a>`
Last not Active	`last`
Next Active	`<a href="#" onclick="{next-link}">next</a>`
Next not Active	`next`
No Match	`no match`

Icon Templates

Icon idle and active path are image URLs to use for custom or default icons. They are used as pins in the map display and referenced by {icon-idle} and {icon-active} in the listing templates. The Popup Box template is only for display inside the popup box that appears when a pin is active, or mouse over. It is only used inside the map view.

Icon Idle Path

Available tokens

All map tokens, field tokens and {index}.

Default template code

```
/DesktopModules/NetismMapExt/Images/Pushpin.aspx?id={index}
```

Icon Acitve Path

Available tokens

All map tokens, field tokens and {index}.

Default template code

```
/DesktopModules/NetismMapExt/Images/Pushpin.aspx?id={index}&selected=1
```

Popup Box

Available tokens

All map tokens, field tokens and processed tokens.

Default template code

```
{title}<hr/>{description}
```

Listing Templates

Result of all listing templates is referenced by {list} in the module view template. If List Alternate Item template is empty, then it would use the List Item template.

List Header

Available tokens

All map tokens.

Default template code

```
{total} location(s) in total<hr/>
```

List Item

Available tokens

All map tokens, field tokens and processed tokens.

Default template code

```
<table><tr> <td valign="top">
  <a href="#" {popup-show} {popup-hide}>
   <img src="{icon-idle}" border="0" id="nmi_{ModuleId}_{Index}" />
  </a>
 </td>
 <td valign="top">
  <a href="{link}" {popup-show} {popup-hide}>{title}</a>
  {link-edit}<br/> *{distance}*
  {address} {city}, {state} {zip} {country}
  {description}
 </td>
</tr></table>
```

List Alternate Item

Available tokens

All map tokens, field tokens and processed tokens.

Default template code

empty

List Separator

Available tokens

All map tokens.

Default template code

```
<hr/>
```

List Footer

Available tokens

All map tokens.

Default template code

```
<br/><hr/>
<a target="_blank" href="http://www.netismsolutions.com/">Netism Map
Extreme</a>
```

9 Search Form

You can define a search form with “Search Tokens” to filter locations dynamically with Ajax, so a full page load is not required. Search tokens can be wrapped with any HTML, Javascript and CSS, allowing you to virtually define a form to filter any fields, with any values, any style, and any layout. Search fields can be set to trigger searches as the user types on a text field, selects an item on a dropdown list, clicks on a checkbox or radio button. It can also be configured to search only when clicking on the button.

The search form, fields and the field values are totally customizable. Single line location input with auto location suggestions and distance filter. A location can be a full address, just a zip code, just a city name, just a state name, just a country name, or even the name of an attraction, or a point of interest.

A search field can be configured to:

Filter	on any data column.
Populate	a list of values from database or a predefined set of values.
Preselect	a default value from a URL parameter or a static value.
Show	the values in a dropdown menu, a radio button, a checkbox, a link, a list of radio buttons, a list of checkboxes or a list of links.

There are 10 types of search tokens, each have their own sets of properties.

9.1 Config

Attribute	Value	Description
type	config	Included on the top of the search form to specify some search behaviors.
presearch	1	The only property currently supported, search when the map first loads.
distance-unit	kilo mile	The unit used for distance calculations.

unit-text	km mi	The unit text used to display in template.

Example
```
{type=config, presearch=1}
{type=config, distance-unit=kilo, unit-text=km}
```

9.2 Text

Attribute	Value	Description
type	text	A text input field.
field		The name of the field to search the text for.
value		Default value, which can be loaded from querystring.

Example
```
{type=text, field=title}
{type=text, field=name, value=store}
{type=text, field=state, value=$state}
```

9.3 Location

Attribute	Value	Description
type	location	A text input field to be used for specifically for location. It can be a full address, just a zip code, just a city name, just a state name, just a country name, or even the name of an attraction or a point of interest etc.
autocomplete	1	Display location suggestion/corrections as location is inputed. It starts on the 3rd entered character.
value		Default value, which can be loaded from querystring.

Example
```
{type=location, autocomplete=1, value=}
{type=location, value=$zip}
```

9.4 Distance

Attribute	Value	Description

type	distance	A dropdown menu containing numeric distance values. The selected value is used to filter the calculated distance based on the entered location.
values		List of predefined values separated by |.
selected		Default value, which can be loaded from querystring.

Example
```
{type=distance, values=0.1|0.5|1|2|5, selected=0.5}
{type=distance, values=10|50|100, selected=$max}
```

9.5 Check

Attribute	Value	Description
type	check	A checkbox.
field		The name of the field to search for the checkbox value.
value		The value of the checkbox.
selected		1 for pre-checking it, 0 otherwise.

Example
```
{type=check, field=country, value=usa, selected=1}
{type=check, field=hasReview, value=true, selected=0}
```

9.6 Radio

Attribute	Value	Description
type	radio	A radio button.
field		The name of the field to search for the radio button value.
value		The value of the radio button.
selected		1 for pre-checking it, 0 otherwise.

Example
```
{type=radio, field=state, value=ny, selected=1}
{type=radio, field=isApproved, value=true, selected=0}
```

9.7 Drop List

Attribute	Value	Description
type	droplist	A dropdown list.

field		The name of the field to search the selected value for.
load	unique	Required.
first		Insert an item on top of the dropdown list.
values		A list of predefined values separated by |, to the dropdown list.
selected		Default value, which can be loaded from querystring.

Example

```
{ type=droplist, field=city, load=unique,
  values=Others, selected=$city }
{ type=droplist, field=state, load=unique,
  first = - pick your state -, values=NY|NJ|PA|CT }
```

9.8 Check List

Attribute	Value	Description
type	checklist	A list of checkboxes.
field		The name of the field to search the checked value for.
load	unique	Required.
first		Insert an item on top of the dropdown list.
values		A list of predefined values separated by |, to the check box list.
selected		Default value, which can be loaded from querystring.
direction	horizontal vertical	Displaying the checkboxes from left to right, or from top to bottom.

Example

```
{ type=checklist, field=city, load=unique,
  values=Others, selected=$city, direction=horizontal }
{ type=checklist, field=state,
  values=NY|NJ|PA|CT, direction=vertical, selected=NY }
```

9.9 Radio List

Attribute	Value	Description
type	radiolist	A list of radio buttons.
field		The name of the field to search the checked value for.

load	unique	Required.
first		Insert an item on top of the dropdown list.
values		A list of predefined values separated by |, to the radio button list.
selected		Default value, which can be loaded from querystring.
direction	horizontal vertical	Displaying the checkboxes from left to right, or from top to bottom.

Example

```
{ type=radiolist, field=city, load=unique,
  values=Others, selected=$city, direction=horizontal }
{ type=radiolist, field=state,
  values=NY|NJ|PA|CT, direction=vertical, selected=NY }
```

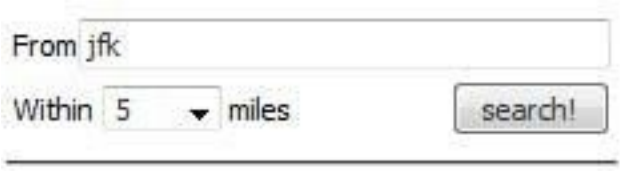
9.10 Button

Attribute	Value	Description
type	button	A button that triggers a search.
value		The label or text of the button.

Example { type=button, value=Find Agents! }

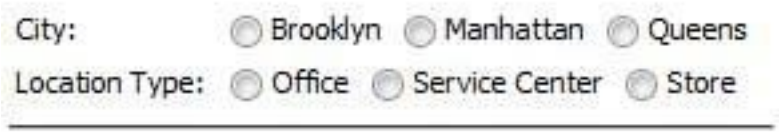
9.11 Example 1 – Default Form

```
{type=config, presearch=1}
<table><tr><td colspan="2">
  From
  {type=location, autocomplete=1, value=$zip, style=width:200px}
</td></tr>
<tr><td>
  Within
  {type=distance, values=0.1|0.5|1|2|5|20|50|100, selected=5}
  Miles
</td><td align="right">
  {type=button, value=Search!}
</td></tr></table>
```



9.12 Example 2 – Simple Form

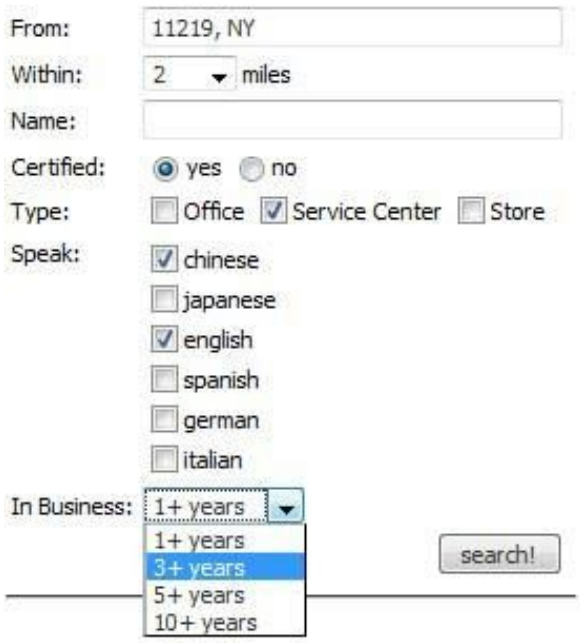
```
{type=config, presearch=1}
<table><tr><td>City:</td><td>
  {type=radiolist, field=city, load=unique, direction=horizontal}
<td></tr>
<tr><td>Location Type:</td><td>
  {type=radiolist, field=ext2, load=unique, direction=horizontal}
<td></tr></table>
```



9.13 Example 3 – Complex Form

```
{type=config, presearch=1}
<table><tr><td>From:</td><td>
  {type=location, autocomplete=1, value=$zip, style=width:200px}
</td></tr>
<tr><td>Within:</td><td>
  {type=distance, values=0.1|0.5|1|2|5|10|20|50|100, selected=2} miles
</td>
<tr><td>Name:</td><td>
  {type=text, field=title, value=, style=width:200px}
<td></tr>
<tr><td>Certified:</td><td>
  {type=radiolist, field=ext1, values=Yes|No, direction=horizontal}
<td></tr>
<tr><td>Type:</td><td>
  {type=checklist, field=ext2, load=unique, direction=horizontal}
<td></tr>
<tr><td valign="top">Speak:</td><td>
  {type=checklist, field=ext3, direction=vertical,
                   values=Chinese|Japanese|English|Spanish|German|Italian }
<td></tr>
<tr><td>In Business:</td><td>
```

```
  {type=droplist, field=ext4,
values=1+ Years|3+ Years|5+ Years|10+ Years}
<td></tr>
<tr><td colspan="2" align="right">
  {type=button, value=Search!}
</td></tr></table>
```



9.14 Note

The power of search tokens combined with your imaginations, you can easily and quickly create flexible and powerful search forms to match all your needs, from searching locations, to filtering categories, to complex combinations.

All search tokens can be quickly and temporarily disabled by adding # in the front.
`{#type=check … }`

Except for config, checklist and radiolist, all search tokens can include other properties, which will be kept in the generated HTML element as their attributes.
`{type=text, class=bigbox, style=width:100px … }`

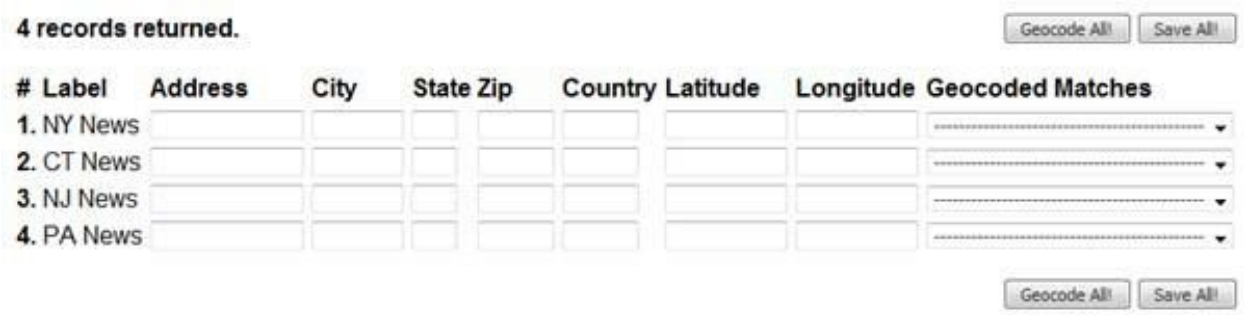
10 Batch Geocoding

The Batch Geocoding page will find and execute a query with the match parameter BatchGeocoding. The returned column with the name Label, Title, Name, or the first text

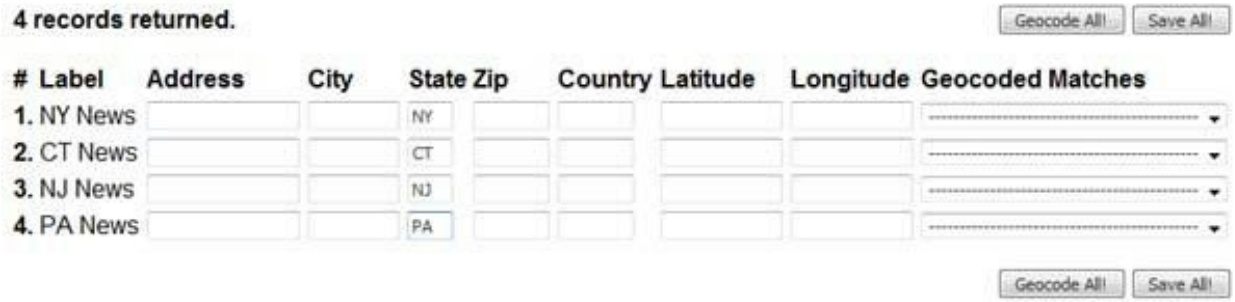
column from the result set will be used for labeling. It would also fill in the Address, City, State, Zip, Country, Latitude and Longitude if any of them have already saved in the database.

Note that this is not an automated or scheduled process. You will need to open and run the batch geocoding form manually.

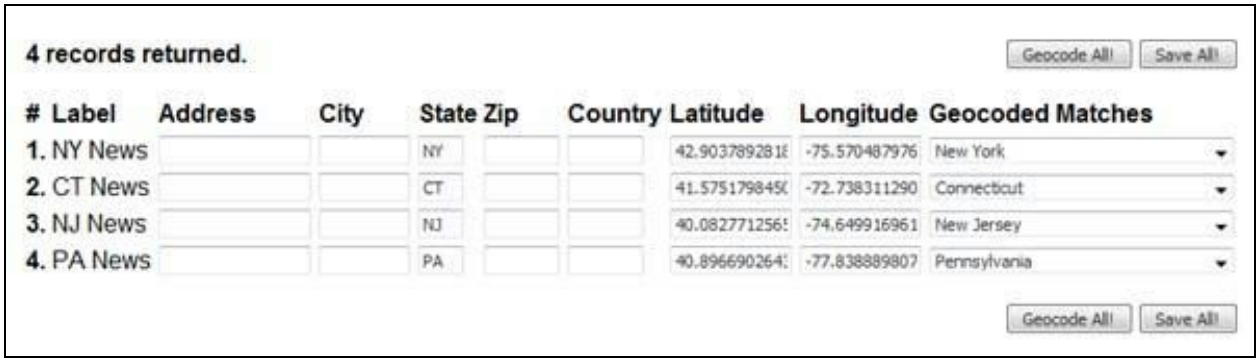
The initial form, loaded with the query we previously built using with the Query Builder.
The repository module has 4 items.



You can simply fill in any parts of the location information, in this case, just the states.



Clicking the “Geocode All!” button would geocode the items based on the address information you fill in. If there are multiple matches, it would load all the matched locations in the Geocoded Matches dropdown. For only one match, it would fill in the Latitude and Longitude.

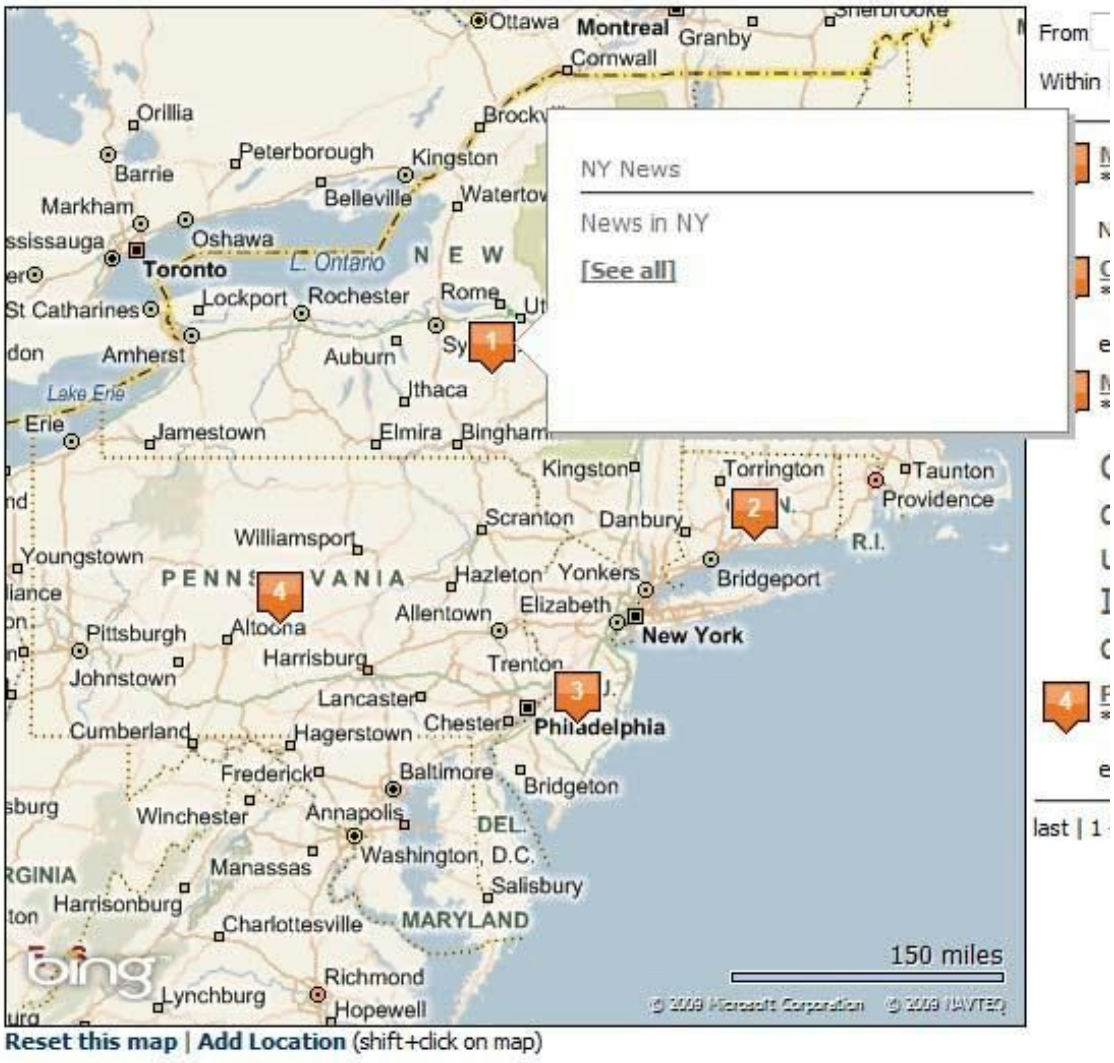


Clicking "Save All!" would save all the information to the database table with update and insert queries like the below. In this case, @TableName would be "grmRepositoryObjects" and the @ID1 would be the ItemID from grmRepositoryObjects. @ID2 and @ID3 would be null.

```
update  Netism_MapExtreme_AttachedLocations
set     Address = @Address, City = @City, State = @State, Zip = @Zip, Country = @Country,
        Latitude = @Latitude, Longitude = @Longitude, Zoom = @Zoom, Iconfile = @Iconfile
where   TableName = @TableName and ID1 = @ID1 and ID2 = @ID2 and ID3 = @ID3

insert  Netism_MapExtreme_AttachedLocations
(TableName, ID1, ID2, ID3, Address, City, State, Zip, Country,
Latitude, Longitude, Zoom, Iconfile)
values  (@TableName, @ID1, @ID2, @ID3, @Address, @City, @State, @Zip, @Country,
@Latitude, @Longitude, @Zoom, @Iconfile)
```

You can set to "Use custom query" under map setting, make sure you have the same query for the default/fallback query. And you should be able to get a map view similar to the below.



This section only applies to the Portal License level, which allows ME to run on only one portal within a DNN installation at a time. Clicking on it and select a portal to enable ME for the DNN installation. For the Host license level, Enterprise license level and Enterprise + Source license level, it is enabled for all portals.

15 Permission

This section only applies to the module when it's set to NOT use custom query. Users other than admin and host will need to be in a role checked on the permission page in order to have access to manipulate map locations.