GEO Locator

V2.0.0

By Denre

Index

License Agreement

It's very simple. Buying a license gives you the right to install one copy of the GEO Locator on one domain. This license is not transferable. Not to a new owner and not to a new domain. If you wish to install the GEO Locator on more than one domain, you will need to buy an extra license.

This is a life-time license and entitles you to any future updates.

Installing the GEO Locator

Prerequisites

libmcrypt

The GEO Locator uses encryption and therefore PHP has to be compiled with libmcrypt. If you're not sure if this library is active, check your phpinfo (Administration/Tools/Host Tools) for mcrypt.

mcrypt

mcrypt support	enabled
mcrypt_filter support	enabled
Version	2.5.8
Api No	20021217
Supported ciphers	cast-128 gost rijndael-128 twofish arcfour cast-256 loki97 rijndael-192 saferplus wake blowfish-compat des rijndael-256 serpent xtea blowfish enigma rc2 tripledes
Supported modes	cbc cfb ctr ecb ncfb nofb ofb stream

Directive	Local Value	Master Value
mcrypt.algorithms_dir	*no value*	*no value*
mcrypt.modes_dir	*no value*	*no value*

For more information about libmcrypt see the following pages of the PHP documentation:

Requirements:

http://php.net/manual/en/mcrypt.requirements.php

Installation:

http://php.net/manual/en/mcrypt.installation.php

World Map

The GEO Locator depends on the World Map module by Boonex being installed. Without this module the GEO Locator won't work properly.

When the World Map module is installed, configure the default location and zoom levels for the different maps.

Be aware; without setting a high zoom level, the map shows the whole world and no change to the map is noticable. A recommended zoom level of 11 is guaranteed to give the desired effect.

Before installation

Installing the GEO Locator can be done in two ways. You can either unzip the “public_html.zip” file in the root of your Dolphin installation, or you can upload the “geo_locator_v2.0.0.zip” file using the built-in FTP functionality Dolphin has to offer.

Be aware; if you decide to use the built-in FTP option, you **FIRST** need to copy the included “BxDolDbGeo.php” to the [Dolphin root]/inc/classes folder.

Installation

If you have carefully followed the previous steps, the GEO Locator will appear in the list of “not installed modules”. Select the module and click install. During the installation a temporary license is installed, which is valid for 14 days from the day of installaton. Without a valid license code, the GEO Locator and your website will stop working!

After installation

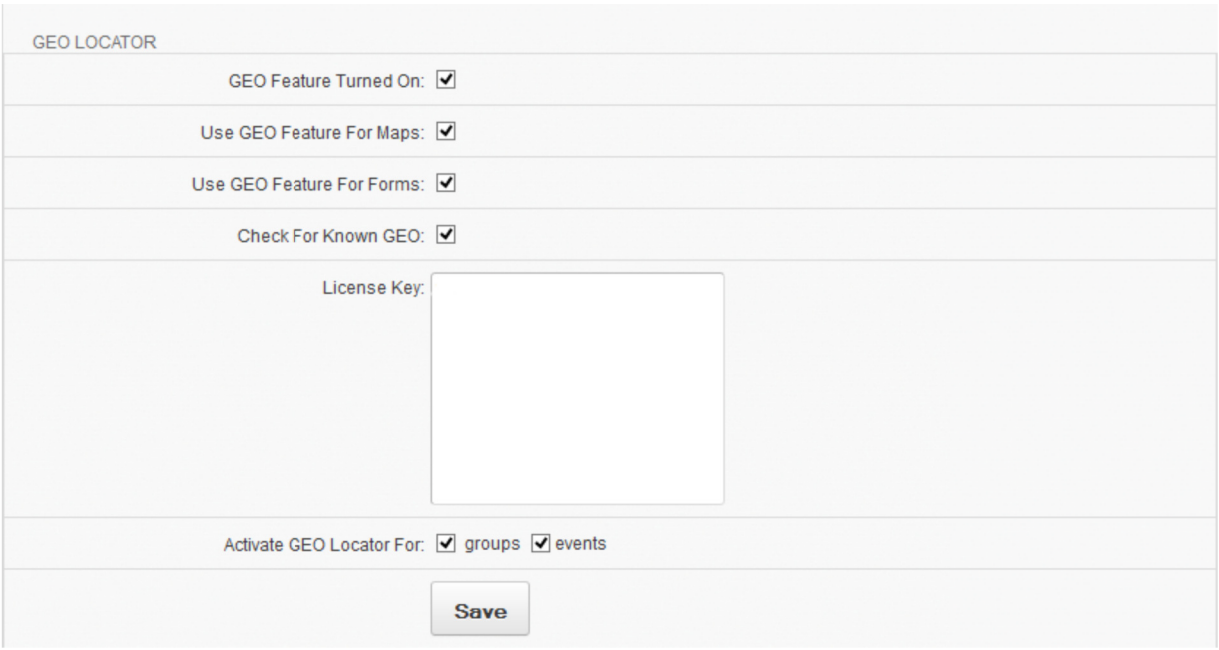
Get license key

The first thing you need to do after installation of the GEO Locator is to send me an email with the (sub)domain name where your copy of the GEO Locator is running and your PayPal transaction id. You than receive a permanent and life-time license key for your copy of the GEO Locator, which you can copy and paste to the GEO Locator settings. Please make sure you do NOT copy any spaces before or after the license key.

Failing to get a valid license for an installed GEO Locator, within 14 days of the first installation of the GEO Locator on that domain, will result in a broken website!

Configure the GEO Locator

Now the GEO Locator is installed and a license key is requested, it's time to configure the GEO Locator and enable it for other modules that support the world map, like groups and events.



For every module that has support for the world map and is installed you can activate the GEO Locator. Once the GEO Locator is activated for a module, new blocks are added to PageBuilder, to replace the existing world map block with. The new blocks have the description “DB GEO”.

Besides the World Map on the homepage there is also a map for the registration page included.

Uninstalling the GEO Locator

Select the “GEO Locator version 2.0.0 by Denre” from the list of installed modules and click “Uninstall”. This will uninstall the module and all references to it. Once the module is uninstalled you can remove the directory [Dolphin root]/modules/denre/geo_locator. If the GEO Locator is the only module of mine that you have installed, you can also remove the directory “denre”. You can now also remove the file [Dolphin root]/inc/classes/BxDolDbGeo.php.

Be aware; do not remove any files before the GEO Locator is properly uninstalled. Deleting files before the GEO Locator is uninstalled will result in a broken website!

Available functions

getMyDistance()

Description

int getMyDistance(int $latTo, int $lngTo, string $unit="")

Returns GEO liniair distance between user and location

Parameters

latTo

– Needs to be a valid latitude

lngTo

– Needs to be a valid longitude

unit

K = Kilometer

M = Miles (default)

N = Nautical Miles

Input is expected in capital K, M or N. The default is Miles

Return Values

Returns a calculated integeter value (int)

getDistance()

Int getDistance(int $latFrom, int $lngFrom, int $latTo, int $lngTo, string $unit="")

Returns GEO liniair distance between 2 points

Parameters

latFrom

– Needs to be a valid latitude

lngFrom

– Needs to be a valid longitude

latTo

– Needs to be a valid latitude

lngTo

– Needs to be a valid longitude

unit

K = Kilometer

M = Miles (default)

N = Nautical Miles

Input is expected in capital K, M or N. The default is Miles

Return Values

Returns a calculated integeter value (int)

getParts()

array getParts()

Returns array of available and active worldmap parts

Parameters

There are no parameters available for this function

Return Values

Returns an two dimensional array of worldmap parts. Each array part contains an associative array

array('profiles' => array())

Key valaues of the returned associative array

Key	Description	Example returned values
Id	Numeric value of row number	1
Part	String containing the name of the wmap part	profiles
title	Language string for the part (plural)	_Profiles
title_singular	Language string for the part (single)	_Profile
icon	Location of the part icon	map_marker_profiles.png
icon_site		User
join_table	Table to join the worldmap table with	Profiles
join_where	WHERE condition for join	AND `p`.`Status` = 'Active'
join_field_id	ID of join field	ID
join_field_country	Field containing Country column for join	Country
join_field_city	Field containing City column for join	City
join_field_state	Field containing State column for join	
join_field_zip	Field containing Zip column for join	Zip
join_field_address	Field containing Address column for join	
join_field_title	Field containing Title column for join	NickName
join_field_uri	Field containing URI column for join	ID
join_field_author	Field containing Author column for join	ID
join_field_privacy	Field containing Privacy info column for join	allow_view_to
permalink		profile.php?ID=
enabled	If the part is active or not	1

getNearbyMe()

array getNearbyMe(string $part, int $distance=10, int $total=50)

Returns an array of IDs and distance nearby the user

Parameters

part

- Valid bx_wmap_parts part

distance

- radius in miles

total

- Max number of entries to return

Return Values

Returns a two dimentional arrray with id's and distances

getNearby()

array getNearby(int $lat, int $lng, string $part, int $distance=10, int $total=50)

Returns array of IDs and distance from any given location

Parameters

lat

- Needs to be a valid latitude

lng

- Needs to be a valid longitude

part

- Valid bx_wmap_parts part

distance

- radius in miles

total

- Max number of entries to return

Return Values

Returns a two dimentional arrray with id's and distances

getHash()

string getHash(int $lat, int $lng)

Returns GEO hashed string from latitude and longitude

Parameters

lat

- Needs to be a valid latitude

lng

- Needs to be a valid longitude

Return values

Returns a GEO hashed string

getfromHash()

array getfromHash(string $hash)

Returns array with latitude, longitude and precision from hash

Parameters

hash

- Valid GEO hashed string

Return Values

Returns an array with latitude, longitude and precision (accuracy) from the hash

Key valaues of the returned associative array

Key	Description	Example returned values
latitude	Integeter value for latitude	51.7579338
longitude	Integeter value for longitude	-0.5638379
precision	The accuracy of the latitude and longitude	1.9515639104739E-17

getGeo()

array getGeo(int $geo, bool $update=false)

Returns GEO information from lat,lng or ip2long($ip)

Parameters

geo

- This needs to be a valid ip2long IP address or valid “latitude, longitude”

update

- Only use update for new location where latitude, longitude and corresponding IP address is known. Every other reason to use the update will polute the location information in the database!

Return Values

Key valaues of the returned associative array

Key	Description	Example returned values
ip	IP address	xxx.xxx.xxx.xxx
country_code	2-letter country codes, as defined in iso 3166	UK
country_name	The name of the Country	United Kingdom
city	String containing City name	London
zipcode	String containing zip code	AA1 1AA
latitude	Integeter value for latitude	51.7579338
longitude	Integeter value for longitude	-0.5638379
latlng	String containing hashed value of the latitude and longitude	gcpxhj8c

getLatLng()

array getLatLng(array $location)

Returns an array with latitude and longitude from location

Parameters

location

- array(

'address' => Valid address,

'city' => Valid city,

'country' => Valid country name

)

Return Values

$aLocation = array('address', 'city', 'country')

getLocation()

Returns location as string from latitude and longitude

getLocation($iLat, $iLng)

getDirectLocation()

Returns location as array

getDirectLocation ($iEntryId, $aPart)

getMyRouteMap()

Returns map of route between user and destination based on destination latitude and longitude

getMyRouteMap($latTo, $lngTo)

getRouteMap()

Returns map of route between two locations based on latitude and longitude

getRouteMap($latFrom, $lngFrom, $latTo, $lngTo)

getMyRoute()

Returns route as array from user location to destination based on latitude and longitude

getMyRoute($latTo, $lngTo)

returns array(

'distance' = array(

'text'

),

'route'

)

getRoute()

Returns route as array between two locations based on latitude and longitude

getRoute($latFrom, $lngFrom)

returns array(

'distance' = array(

'text'

),

'route'

)

isValidLongitude()

Checks validity of longitude and returns true or false

isValidLongitude($longitude)

isValidLatitude()

Checks validity of latitude and returns true or false

isValidLatitude($latitude)

Examples

Including the GEO Locator in to your own module

The following example is to include the GEO Locator and get some default values

```
//import the class
bx_import('BxDolDbGeo');

//create a new object for use
$oDbGeo = new BxDolDbGeo();
```

$oDbGeo contains:

```
 array(
        'lat' => (int),
        'lng' => (int),
        'zip' => string,
        'city' => string,
        'country_name' => string
        'country_code' => string
)
```

Get list of modules that have built-in support for the Boonex World Map

```
$aDbGeoParts = $this->oDbGeo->getParts();
```

Add a route map to modules with World Map Support

```
// The example uses the events module
//World Map Part for our Module
$sPart = 'events';

//fictive entry ID
$iEntryId = 1;

//check if our module supports the World Map
if (!isset($aDbGeoParts[$sPart]))
        return '';
```

```
        //If available, get the location information for the event stored in the database
        if($r = $oDbGeo->getDirectLocation($iEntryId, $this->aDbGeoParts[$sPart]))
        {
                //see if we can get an array of the route
                if($aRouteTxt = $oDbGeo->getMyRoute($r['lat'], $r['lng']))
                {
                        //Prepare our template
                        foreach($aRouteTxt['route'] as $iId => $sRouteTxt)
                        {
                                $aVars['bx_repeat:route_txt'][] = array (
                                        'line' => $sRouteTxt,
                                );
                        }

                        //Get the distance
                        $aVars['distance'] = $aRouteTxt['distance']['text'];

                        //parse the template
                        $tRouteTxt = $this->_oTemplate->parseHtmlByName('routetxt.html',
$aVars);
                }
        }
```

Add a route map to modules without World Map Support

```
//fictive entry ID
$iEntryId = 1;

//Get location using a query to our module entry
//$aLocation = array('address', 'city', 'country')
$aLocation = $this->_oDb->getLocation($iEntryId);

//Get latitude and longitude of the location
$LatLng = $oDbGeo->getLatLng($aLocation);



//see if we can get an array of the route
if($aRouteTxt = $this->oDbGeo->getMyRoute($LatLng['lat'], $LatLng['lng']))
{
	//Prepare our template
	foreach($aRouteTxt['route'] as $iId => $sRouteTxt)
	{
		$aVars['bx_repeat:route_txt'][] = array (
			'line' => $sRouteTxt,
		);
	}

	//Get the distance
	$aVars['distance'] = $aRouteTxt['distance']['text'];

	//parse the template
	//see template routetxt.html for this example
	$tRouteTxt = $this->_oTemplate->parseHtmlByName('routetxt.html', $aVars);
}
```