Google Maps V3 API

CodeIgniter Library

Author: BIOSTALL (Steve Marks)

Website: http://biostall.com

Email: info@biostall.com

Introduction

This CodeIgniter library provides an easy way to display simple maps within an application/website using the Google Maps V3 API. It allows maps to be shown, including customisable markers, polylines, polygons and more with just a few lines of code. Directions can also be drawn, both onto the map, and textual directions written to an element on the page.

Installation

In order to use the library you will need to download the Googlemaps.php file and place it in the 'libraries' directory of your application folder. Following the demonstrations and documentation below you will then be able to begin creating maps right away.

The Basics

Once the installation instructions above are complete there are just two things we need to do to in order to create a map. The first is to amend our controller in order to initialize and customise our output, and the second is to include two lines of code in our view where we want the map to be displayed:

The Controller

The example below shows how to create a very simple map with all the default controls/properties and no overlays/markers:

```
// Load the library
$this->load->library('googlemaps');

// Initialize our map. Here you can also pass in additional parameters for customising the map (see below)
$this->googlemaps->initialize();

// Create the map. This will return the Javascript to be included in our pages <head></head> section and the HTML code to be
// placed where we want the map to appear.
$data['map'] = $this->googlemaps->create_map();

// Load our view, passing the map data that has just been created
$this->load->view('my_view', $data);
```

The View

There are two things we need to add to our view. The first is the Javascript which should be placed in the <head> section of your website as follows:

```
<head>
<?php echo $map['js']; ?>
</head>
```

The next thing is the actual map. This should be placed where you want the map to appear as follows:

```
<?php echo $map['html']; ?>
```

Customising the Map

The library allows you to adjust the way your map appears by passing in additional values in a $config array to $this->googlemaps->initialize($config). These are as follows:

Name	Type	Default	Possible Values	Description
$adsense	boolean	FALSE	TRUE, FALSE	Whether Google Adsense For Content should be enabled
$adsenseChannelNumber	string			The Adsense channel number for tracking the performance of this AdUnit
$adsenseFormat	string	HALF_BANNER	"BANNER", "BUTTON", "HALF_BANNER", "LARGE_RECTANGLE", "LEADERBOARD", "MEDIUM_RECTANGLE", "SKYSCRAPER", "SMALL_RECTANGLE", "SMALL_SQUARE", "SQUARE", "VERTICAL_BANNER", "WIDE_SKYSCRAPER"	The format of the AdUnit
$adsensePosition	string	TOP_CENTER		The position of the AdUnit
$adsensePublisherID	string			Your Google AdSense publisher ID
$center	string	"37.4419, -122.1419"	A latitude/longitude coordinate OR an address. If an address is supplied the Google geocoding service will be used to return a lat/long.	Sets the default center location of the map
$directions	boolean	FALSE		Whether or not the map will be used to show directions
$directionsStart	string		A latitude/longitude coordinate OR an address. If an address is supplied the Google geocoding service will be used to return a lat/long.	The starting location of the directions
$directionsEnd	string		A latitude/longitude coordinate OR an address. If an address is supplied the Google geocoding service will be used to return a lat/long.	The destination point of the directions
$directionsDivID	string			An element's ID on the page where textual directions will be output to. Leave blank if not required
$directionsMode	string	“DRIVING”	“DRIVING”, “WALKING”, “BICYCLING” (US only)	The vehicle/mode of transport to show directions for
$directionsAvoidTolls	boolean	FALSE	TRUE, FALSE	Whether or not directions should avoid tolls
$directionsAvoidHighways	boolean	FALSE	TRUE, FALSE	Whether or not directions should avoid highways
$disableDefaultUI	boolean	FALSE	TRUE, FALSE	If set to TRUE will hide the default controls (ie. zoom, scale)
$disableMapTypeControl	boolean	FALSE	TRUE, FALSE	If set to TRUE will hide the MapType control (ie. Map, Satellite, Hybrid, Terrain)

$disableNavigationControl	boolean	FALSE	TRUE, FALSE	If set to TRUE will hide the Navigation control (ie. zoom in/out, pan)
$disableScaleControl	boolean	FALSE	TRUE, FALSE	If set to TRUE will hide the Scale control
$disableDoubleClickZoom	boolean	FALSE	TRUE, FALSE	If set to TRUE will disable zooming when a double click occurs
$draggable	boolean	TRUE	TRUE, FALSE	If set to FALSE will prevent the map from being dragged around
$draggableCursor	string			The name or url of the cursor to display on a draggable object
$draggingCursor	string			The name or url of the cursor to display when an object is dragging
$navigationControlPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Navigation control
$keyboardShortcuts	boolean	TRUE	TRUE, FALSE	If set to FALSE will disable to map being controlled via the keyboard
$jsfile	string			Set this to the path of an external JS file if you wish the Javascript to be placed in a file rather than output directly into the <head></head> section. The library will try to create the file if it does not exist already. Please ensure the destination file is writeable
$map_div_id	string	“map_canvas”		The ID of the that is output which contains the map
$map_height	string	“450px”		The height of the map container. Any units (ie ‘px’) can be used. If no units are provided ‘px’ will be presumed
$map_name	string	“map”		The JS reference to the map. Currently not used but to be used in the future when multiple maps are supported
$map_type	string	“ROADMAP”	“HYBRID”, “ROADMAP”, “SATELLITE”, “TERRAIN”	The default MapType
$map_width	string	“100%”		The width of the map container. Any units (ie ‘px’) can be used. If no units are provided ‘px’ will be presumed
$mapTypeControlPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the MapType control
$region	string			Country code top-level domain (eg “uk”) within which to search. Useful if supplying addresses rather than lat/longs
$scaleControlPosition	string		“BOTTOM”, “BOTTOM_LEFT”,	The position of the Scale

			“BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	control
$scrollwheel	boolean	TRUE	TRUE, FALSE	If set to FALSE will disable zooming by scrolling of the mouse wheel
$sensor	boolean	FALSE	TRUE, FALSE	Set to TRUE if being used on a device that can detect a users location
$zoom	string	“13”	“0” (zoomed out) – “18” (zoomed in), “auto”	The default zoom level of the map. If set to “auto” will auto-zoom/center to fit in all visible markers. If “auto”, also overrides the $center parameter

Adding Markers

The library also allows you to add multiple markers to the map at specified positions. To add a single marker we can add the following code BEFORE the create_map() function is called:

```
// Set the marker parameters as an empty array. Especially important if we are using multiple markers
$marker = array();

// Specify an address or lat/long for where the marker should appear.
$marker[' position '] = 'Crescent Park, Palo Alto';

// Once all the marker parameters have been specified lets add the marker to our map
this->googlemaps->add_marker($marker);
```

To create multiple markers simply duplicate the above code the required amount of times.

Like the map itself, we can also specify a number of parameters for individual markers to change how and where they appear. These parameters are as follows:

Name	Type	Default	Possible Values	Description
$position	string		A latitude/longitude coordinate OR an address. If an address is supplied the Google geocoding service will be used to return a lat/long.	The position at which the marker will appear
$infowindow_content	string			If not blank, creates an infowindow (aka bubble) with the content provided. Can be plain text or HTML
$animation	string		“DROP”, “BOUNCE”	Animate the marker so it exhibits dynamic movement
$clickable	boolean	TRUE	TRUE, FALSE	Defines if the marker is clickable
$cursor	string			The name or url of the cursor to display on hover
$draggable	boolean	FALSE	TRUE, FALSE	Defines if the marker is draggable
$flat	boolean	FALSE	TRUE, FALSE	If set to TRUE will no display a shadow beneath the icon
$icon	string			The name or url of the icon to use for the marker
$onclick	string			JavaScript performed when a marker is clicked
$ondragstart	string			JavaScript performed when a marker is started to be dragged
$ondragend	string			JavaScript performed when a draggable marker is dropped
$shadow	string			The name or url of the icon’s shadow
$title	string			The tooltip text to show on hover
$visible	boolean	TRUE	TRUE, FALSE	Defines if the marker is visible by default
$zIndex	int			The zIndex of the marker. If two markers overlap, the marker with the higher $zIndex will appear on top

Adding Polylines

The library also allows you to add polylines to the map at specified positions. To add a single polyline we can add the following code BEFORE the create_map() function is called:

```
// Set the polyline parameters as an empty array. Especially important if we are using multiple polylines
$polyline = array();

// Specify an array of addresses or lat/longs for where the polyline points should appear.
$polyline['points'] = array('37.429, -122.1319', 'Crescent Park, Palo Alto', '37.4419, -122.1219');

// Once all the polyline parameters have been specified lets add the polyline to our map
$this->googlemaps->add_polyline($polyline);
```

To create multiple polylines simply duplicate the above code the required amount of times.

We can also specify a number of parameters for individual polylines to change how and where they appear. These parameters are as follows:

Name	Type	Default	Possible Values	Description
$points	array		An array of latitude/longitude coordinates OR addresses, or a mixture of both. If an address is supplied the Google geocoding service will be used to return a lat/long.	The position at which the polyline points will appear
$strokeColor	string	“#FF0000”	A hex colour value	The colour of the polyline
$strokeOpacity	string	“1.0”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the polyline
$strokeWeight	string	“2”		The thickness of the polyline

Adding Polygons

The library also allows you to add polygons to the map at specified positions. To add a single polygon we can add the following code BEFORE the create_map() function is called:

```
// Set the polygon parameters as an empty array. Especially important if we are using multiple polygons
$polygon = array();

// Specify an array of addresses or lat/longs for where the polygon points should appear.
// NOTE: The first and last elements in the array should be the same to complete the polygon
$polygon['points'] = array('37.425, -122.1321', '37.4422, -122.1622', '37.4412, -122.1322', '37.425, -122.1021');

// Once all the polygon parameters have been specified lets add the polygon to our map
$this->googlemaps->add_polygon($polygon);
```

To create multiple polygons simply duplicate the above code the required amount of times.

We can also specify a number of parameters for individual polygons to change how and where they appear. These parameters are as follows:

Name	Type	Default	Possible Values	Description
$points	array		An array of latitude/longitude coordinates OR addresses, or a mixture of both. If an address is supplied the Google geocoding service will be used to return a lat/long.	The position at which the polygon points will appear. NOTE: The first and last elements of the array must be the same
$strokeColor	string	“#FF0000”	A hex colour value	The colour of the polygon border
$strokeOpacity	string	“0.8”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the polygon border
$strokeWeight	string	“2”		The thickness of the polygon border
$fillColor	string	“#FF0000”	A hex colour value	The colour of the polygon
$fillOpacity	string	“0.3”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the polygon

Adding Circles

The library also allows you to add circles to the map at specified positions. To add a single circle we can add the following code BEFORE the create_map() function is called:

```
// Set the circle parameters as an empty array. Especially important if we are using multiple circles
$circle = array();

// Specify the center and radius (in metres) of the circle
$circle['center'] = '37.459, -122.1319';
$circle['radius'] = '1000';

// Once all the circle parameters have been specified lets add the circle to our map
$this->googlemaps->add_circle($circle);
```

To create multiple circles simply duplicate the above code the required amount of times.

We can also specify a number of parameters for individual circles to change how and where they appear. These parameters are as follows:

Name	Type	Default	Possible Values	Description
$center	string		A latitude/longitude coordinate OR an address. If an address is supplied the Google geocoding service will be used to return a lat/long.	The position at which the circle will appear
$radius	int	0		The circle radius (in metres).
$strokeColor	string	“#FF0000”	A hex colour value	The colour of the circle border
$strokeOpacity	string	“1.0”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the circle border
$strokeWeight	string	“2”		The thickness of the circle border
$fillColor	string	“#FF0000”	A hex colour value	The colour of the circle
$fillOpacity	string	“0.3”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the circle

Need Help?

To leave feedback, ask questions or report bugs please contact me at info@biostall.com or leave a comment at http://biostall.com.