Google Maps V3 API

CodeIgniter Library

Author: BIOSTALL (Steve Marks)

Website: http://biostall.com

Link: http://biostall.com/codeigniter-google-maps-v3-api-library

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, circles, rectangles, ground overlays 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. The library also integrates with Google Places to show places of interest.

Installation

In order to use the library you will need to download the Googlemaps.php file and place it in your 'application/libraries' directory. 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
$backgroundColor	string		A hex colour value	The background color shown when tiles have not yet loaded as the user pans
$bicyclingOverlay	boolean	FALSE	TRUE, FALSE	If set to TRUE will overlay bicycling information (ie. bike paths and suggested routes) onto the map by default
$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. Other possible value is “auto” (see description).	Sets the default center location of the map. You can also center on the current users location using available geocoding services. Simply set this to “auto”.
$cluster	boolean	FALSE	TRUE, FALSE	Whether to cluster markers
$clusterGridSize	integer	60		The grid size of a cluster in pixels
$clusterMaxZoom	integer			The maximum zoom level that a marker can be part of a cluster
$clusterZoomOnClick	boolean	TRUE	TRUE, FALSE	Whether the default behaviour of clicking on a cluster is to zoom into it
$clusterAverageCenter	boolean	FALSE	TRUE, FALSE	Whether the center of each cluster should be the average of all markers in the cluster
$clusterMinimumClusterSize	integer	2		The minimum number of markers to be in a cluster before the markers are hidden and a count is shown
$directions	boolean	FALSE		Whether or not the map will be used to show directions
$directionsChanged	string			JavaScript to perform when directions are dragged. $directionsDraggable must be

				set to true in order to use this parameter
$directionsDraggable	boolean	FALSE	TRUE, FALSE	Whether or not directions on the map are draggable
$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
$disableStreetViewControl	boolean	FALSE	TRUE, FALSE	If set to TRUE will hide the Street View 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
$geocodeCaching	boolean	FALSE	TRUE, FALSE	If set to TRUE will cache any geocode requests made when an address is used instead of a lat/long. Requires database table to be created (see below).
$https	boolean	FALSE	TRUE, FALSE	If set to TRUE will load the Google Maps JavaScript API over HTTPS, allowing you to utilize the API within your HTTPS secure application

$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
$kmlLayerURL	string			A URL to publicly available KML or GeoRSS data for displaying geographic information
$kmlLayerPreserveViewport	boolean	FALSE	TRUE, FALSE	Specifies whether the map should be adjusted to the bounds of the KmlLayer's contents. By default the map is zoomed and positioned to show the entirety of the layer's contents
$loadAsynchronously	boolean	FALSE	TRUE, FALSE	Load the map and API asynchronously once the page has loaded
$map_div_id	string	“map_canvas”		The ID of the output that 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”, “STREET”	The default MapType.
$map_types_available	array		"HYBRID", "ROADMAP", "SATELLITE", "TERRAIN"	An array of map types to show available for selection on the map
$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
$mapTypeControlStyle	string		"DROPDOWN_MENU", "HORIZONTAL_BAR"	The style of the MapType control
$minzoom	integer			The minimum zoom level which will be displayed on the map
$maxzoom	integer			The maximum zoom level which will be displayed on the map

$minifyJS	boolean	FALSE	TRUE, FALSE	If set to TRUE will run the JavaScript created by the library through Jsmin.php (this file, available in the repository, and PHP5+ required) to minify the code produced
$noClear	boolean	FALSE	TRUE, FALSE	If TRUE do not clear the contents of the map div
$onboundschanged	string			The JavaScript action to perform when the viewport bounds have changed
$oncenterchanged	string			The JavaScript action to perform when the map center property changes
$onclick	string			The JavaScript action to perform when the map is clicked
$ondblclick	string			The JavaScript action to perform when the map is double-clicked
$ondrag	string			The JavaScript action to perform while the map is dragged
$ondragend	string			The JavaScript action to perform when the user stops dragging the map
$ondragstart	string			The JavaScript action to perform when the user starts dragging the map
$onidle	string			The JavaScript action to perform when the map becomes idle after panning or zooming
$onload	string			The JavaScript action to perform when the map first loads. This library hi-jacks the window.load event so add any bespoke code using this option
$onmousemove	string			The JavaScript action to perform when the user's mouse moves over the map container
$onmouseout	string			The JavaScript action to perform when the user's mouse exits the map container
$onmouseover	string			The JavaScript action to perform when the user's mouse enters the map container
$onresize	string			The JavaScript action to perform when the maps div changes size
$onrightclick	string			The JavaScript action to perform when the map is right-clicked
$ontilesloaded	string			The JavaScript action to perform when the visible tiles have finished loading
$onzoomchanged	string			The JavaScript action to perform when the maps zoom property changes
$panoramio	boolean	FALSE	TRUE, FALSE	If TRUE will add photos from Panoramio as a layer to your maps as a series of large and small photo icons

$panoramioTag	string			Restrict the set of Panoramio photos shown to those matching a certain textual tag
$panoramioUser	string		Any valid Panoramio User ID	Restrict the set of Panoramio photos shown to those matching a particular user
$places	boolean	FALSE	TRUE, FALSE	Whether or not the map will be used to show places
$placesLocation	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.	A point on the map if the search for places is based around a central point
$placesRadius	integer	0		The radius (in meters) if search is based on a central position
$placesLocationSW	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.	If preferring to search within bounds the South-West position
$placesLocationNE	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.	If preferring to search within bounds the North-East position
$placesTypes	array		For a full list of accepted types see: http://code.google.com/apis/maps/documentation/places/supported_types.html	An array of types of places to search for
$placesName	string			A term to be matched against when searching for places to display on the map
$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”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Scale 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
$streetViewAddressControl	boolean	TRUE	TRUE, FALSE	If set to FALSE will hide the Address control
$streetViewAddressPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Address control, eg. 'TOP_LEFT'
$streetViewControlPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Street View control
$streetViewCloseButton	boolean	FALSE	TRUE, FALSE	If set to TRUE will show the close button in the top right. The

				close button allows users to return to the aerial map
$streetViewLinksControl	boolean	TRUE	TRUE, FALSE	If set to FALSE will hide the Links control
$streetViewPanControl	boolean	TRUE	TRUE, FALSE	If set to FALSE will hide the Pan control
$streetViewPanPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Scale control, eg. 'TOP_RIGHT'
$streetViewPovHeading	integer	0	0 to 359	The Street View camera heading in degrees relative to true north. True north is 0, east is 90, south is 180, west is 270
$streetViewPovPitch	integer	0	90 to -90	The Street View camera pitch in degrees, relative to the street view vehicle. Directly upwards is 90, Directly downwards is -90.
$streetViewPovZoom	integer	0		The Street View zoom level. Fully zoomed-out is level 0, zooming in increases the zoom level.
$streetViewZoomControl	boolean	TRUE	TRUE, FALSE	If set to FALSE will hide the Zoom control
$streetViewZoomPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Scale control, eg. 'TOP_RIGHT'
$streetViewZoomStyle	string		"SMALL", "LARGE"	The size of the Street View zoom control.
$styles	array			An array of styles used to colour aspects of the map and turn points of interest on and off. Gets converted to a JSON array. See the website demonstration for clarification on how this should be implemented.
$stylesAsMapTypes	boolean	FALSE	TRUE, FALSE	If applying styles, whether to apply them to the default map or add them as additional map types
$stylesAsMapTypesDefault	string			If $stylesAsMapTypes is true the default style. Should contain the 'Name' of the style
$trafficOverlay	boolean	FALSE	TRUE, FALSE	If set to TRUE will overlay traffic information onto the map by default
$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
$zoomControlPosition	string		“BOTTOM”, “BOTTOM_LEFT”, “BOTTOM_RIGHT”, “LEFT”, “RIGHT”, “TOP”, “TOP_LEFT”, “TOP_RIGHT”	The position of the Zoom control
$zoomControlStyle	string		"SMALL", "LARGE"	The style of the Zoom control

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
$ondblclick	string			JavaScript performed when a marker is double-clicked
$ondrag	string			JavaScript repeatedly performed while the marker is being dragged
$ondragstart	string			JavaScript performed when a marker is started to be dragged
$ondragend	string			JavaScript performed when a draggable marker is dropped
$onmousedown	string			JavaScript performed when a mousedown event occurs on a marker
$onmouseout	string			JavaScript performed when the mouse leaves the area of the marker icon

$onmouseover	string			JavaScript performed when the mouse enters the area of the marker icon
$onmouseup	string			JavaScript performed when a mouseup event occurs on a marker
$onrightclick	string			JavaScript performed when a right-click occurs on a marker
$raiseondrag	TRUE		TRUE, FALSE	If FALSE, disables the raising and lowering of the icon when a marker is being dragged
$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
$clickable	boolean	TRUE	TRUE, FALSE	Defines if the polyline is clickable
$onclick	string			JavaScript performed when a polyline is clicked
$ondblclick	string			JavaScript performed when a polyline is double-clicked
$onmousedown	string			JavaScript performed when a mousedown event occurs on a polyline
$onmousemove	string			JavaScript performed when the mouse moves in the area of the polyline
$onmouseout	string			JavaScript performed when the mouse leaves the area of the polyline
$onmouseover	string			JavaScript performed when the mouse enters the area of the polyline
$onmouseup	string			JavaScript performed when a mouseup event occurs on a polyline
$onrightclick	string			JavaScript performed when a right-click occurs on a polyline
$zIndex	int			The zIndex of the polyline. If two polylines overlap, the polyline with the higher $zIndex will appear on top

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
$clickable	boolean	TRUE	TRUE, FALSE	Defines if the polygon is clickable
$onclick	string			JavaScript performed when a polygon is clicked
$ondblclick	string			JavaScript performed when a polygon is double-clicked
$onmousedown	string			JavaScript performed when a mousedown event occurs on a polygon
$onmousemove	string			JavaScript performed when the mouse moves in the area of the polygon
$onmouseout	string			JavaScript performed when the mouse leaves the area of the polygon
$onmouseover	string			JavaScript performed when the mouse enters the area of the polygon
$onmouseup	string			JavaScript performed when a mouseup event occurs on a polygon

$onrightclick	string			JavaScript performed when a right-click occurs on a polygon
$zIndex	int			The zIndex of the polygon. If two polygons overlap, the polygon with the higher $zIndex will appear on top

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	integer	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
$clickable	boolean	TRUE	TRUE, FALSE	Defines if the circle is clickable
$onclick	string			JavaScript performed when a circle is clicked
$ondblclick	string			JavaScript performed when a circle is double-clicked
$onmousedown	string			JavaScript performed when a mousedown event occurs on a circle
$onmousemove	string			JavaScript performed when the mouse moves in the area of the circle
$onmouseout	string			JavaScript performed when the mouse leaves the area of the circle
$onmouseover	string			JavaScript performed when the mouse enters the area of the circle
$onmouseup	string			JavaScript performed when a mouseup event occurs on a circle

$onrightclick	string			JavaScript performed when a right-click occurs on a circle
$zIndex	int			The zIndex of the circle. If two circles overlap, the circle with the higher $zIndex will appear on top

Adding Rectangles

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

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

// Specify the bounds (south-west and north-east positions) of the rectangle
$rectangle['positionSW'] = '37.459, -122.1319';
$rectangle['positionNE'] = '37.459, -122.2244';

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

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

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

Name	Type	Default	Possible Values	Description
$positionSW	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 South-West position at which the rectangle will appear
$positionNE	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 North-East position at which the rectangle will appear
$strokeColor	string	“#FF0000”	A hex colour value	The colour of the rectangle border
$strokeOpacity	string	“1.0”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the rectangle border
$strokeWeight	string	“2”		The thickness of the rectangle border
$fillColor	string	“#FF0000”	A hex colour value	The colour of the rectangle
$fillOpacity	string	“0.3”	0 (transparent) – 1.0 (solid/opaque)	The opacity of the rectangle
$clickable	boolean	TRUE	TRUE, FALSE	Defines if the rectangle is clickable
$onclick	string			JavaScript performed when a rectangle is clicked
$ondblclick	string			JavaScript performed when a rectangle is double-clicked
$onmousedown	string			JavaScript performed when a mousedown event occurs on a rectangle
$onmousemove	string			JavaScript performed when the mouse moves in the area of the rectangle
$onmouseout	string			JavaScript performed when the mouse leaves the area of the rectangle
$onmouseover	string			JavaScript performed when the mouse enters the area of the rectangle

$onmouseup	string			JavaScript performed when a mouseup event occurs on a rectangle
$onrightclick	string			JavaScript performed when a right-click occurs on a rectangle
$zIndex	int			The zIndex of the rectangle. If two rectangles overlap, the rectangle with the higher $zIndex will appear on top

Adding Ground Overlays

The library also allows you to add ground overlays (ie. images) to the map at specified positions. The image will automatically resize to match the South-West and North-East positions provided. To add a single ground overlay we can add the following code BEFORE the create_map() function is called:

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

// Specify the bounds (south-west and north-east positions) of the ground overlay
$overlay['image'] = 'http://www.mydomain.com/images/test.jpg';
$overlay['positionSW'] = '37.459, -122.1319';
$overlay['positionNE'] = '37.459, -122.2244';

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

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

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

Name	Type	Default	Possible Values	Description
$image	string		A valid URL	A URL to the image that should be used as the overlay
$positionSW	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 South-West position at which the overlay will appear
$positionNE	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 North-East position at which the overlay will appear
$clickable	boolean	TRUE	TRUE, FALSE	Defines if the overlay is clickable
$onclick	string			JavaScript performed when an overlay is clicked

Caching Geocoding Requests

A geocode request is made from within the library whenever an address or textual location is used instead of a latitude/longitude co-ordinate. Whether it be centering the map or adding a marker, each time one of these requests is required the library goes off to a service provided by Google and obtains the required information.

There are two benefits to caching these types of request:

1) Speed. Getting the latitude and longitude from a database for a certain location will be quicker than going off to ask Google each time.
2) Request Limitations. Google allow you make 2,500 geocode requests per day. Caching helps massively if you're using this library on a high traffic site, or are frequently requesting details for the same location.

To use the caching option provided by the library you need an established database connection and to follow the steps below:

1) Create the database table. Simply copy and paste the SQL statement below to do this:

```
CREATE TABLE IF NOT EXISTS `geocoding` (
        `address` varchar(255) NOT NULL DEFAULT '',
        `latitude` float DEFAULT NULL,
        `longitude` float DEFAULT NULL,
        PRIMARY KEY (`address`)
)
```

2) Activate the library configuration option $geocodeCaching by setting it to TRUE:

```
$config['geocodeCaching'] = TRUE;
$this->googlemaps->initialize($config);
```

Minifying Javascript

The result of this library is a bunch of Javascript that, based on other options, is either output into the HTML or wrote to a JS file. When adding lots of markers, overlays and other functionality to the map this JavaScript can get quite large, in which case we recommend that you minify this code.

To perform minimisation of the Javascript produced by the library you will need to download the file Jsmin.php that is available in the repository and place it in the 'libraries' folder of your application directory.

Then simply activate this feature by setting the library configuration option $minifyJS by setting it to TRUE:

```
$config['minifyJS'] = TRUE;
$this->googlemaps->initialize($config);
```

Adding Markers From Database Co-ordinates

A common question I get asked is “How can I loop through a set of co-ordinates and add a marker to the map for each record?”. As a result I have added this section of the documentation to provide an example of how to accomplish this.

The Database

For the scenario listed below we'll imagine we've got a very simple database table with just two columns; *lat* and *lng*. The MySQL for creating this table would look something like so:

```
CREATE TABLE `coords` (
        `lat` float DEFAULT NULL,
        `lng` float DEFAULT NULL
)
```

No doubt your table will be named different, have different named columns and more fields. As a result please ensure these differences are reflected if copying the code below.

The Model

Now that we've got our database set up we'll need a way to get these co-ordinates out so that we can use them later on on the map. In sticking to the conventional MVC (Model-View-Controller) structure we'll do this in a model.

For this example we'll create a model called 'map_model.php' that might look something like so:

```
<?php
class Map_model extends CI_Model {
   function __construct()
   {
      parent::__construct();
   }

   function get_coordinates()
   {

      $return = array();

      $this->db->select("lat,lng");
      $this->db->from("coords");
      $query = $this->db->get();

      if ($query->num_rows()>0) {
         foreach ($query->result() as $row) {
            array_push($return, $row);
         }
      }

      return $return;

   }
}
```

In the model above we are getting all latitude and longitude coordinates from our table. We're then executing the query and looping through the results, pushing each one to an array variable called *$return*. The resulting output is an array that we'll use in a little while containing all the data we need to start plotting markers.

The Controller

Nearly there... The final step is to take the array of co-ordinates that we generated in our model and plot them on the map using this library. Our sample controller, in this case called 'map.php', would look something like so:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Map extends CI_Controller {

   function __construct()
   {
      parent::__construct();
   }

   function index()
   {
      // Load the library
      $this->load->library('googlemaps');

      // Load our model
      $this->load->model('map_model', '', TRUE);

      // Initialize the map, passing through any parameters
      $config['center'] = '1600 Amphitheatre Parkway in Mountain View, Santa Clara County, California';
      $config['zoom'] = "auto";
      $this->googlemaps->initialize($config);

      // Get the co-ordinates from the database using our model
      $coords = $this->map_model->get_coordinates();

      // Loop through the coordinates we obtained above and add them to the map
      foreach ($coords as $coordinate) {
         $marker = array();
         $marker['position'] = $coordinate->lat.','.$coordinate->lng;
         $this->googlemaps->add_marker($marker);
      }

      // Create the map
      $data = array();
      $data['map'] = $this->googlemaps->create_map();

      // Load our view, passing through the map data
      $this->load->view('map_view', $data);

   }

}
```

All that's left to do now is to generate your view. I'll leave that bit to you ;)

Tracking Markers Externally

Aside from the library returning the Javascript and HTML required for the map, there is a third set of data returned as a result of calling the create_map() function.

This array, called 'markers', contains information about the markers that were added to the map. This information includes the markers' ID for use in custom JavaScript, the latitude and longitude, and the title if one was set.

An example for obtaining this array can be seen below:

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

$this->googlemaps->initialize($config);

$marker = array();
$marker['position'] = '1600 Amphitheatre Parkway in Mountain View, Santa Clara County, California';
$marker['title'] = 'A marker title';
$this->googlemaps->add_marker($marker);

$data['map'] = $this->googlemaps->create_map();

print_r($data['map']['markers']);
```

The above would output the marker information something like so:

```
Array(
   [marker_0] => Array (
         [latitude] => 37.4213068
         [longitude] => -122.08529
         [title] => A marker title
      )
)
```

In the above example 'marker_0' is the ID of the marker should you wish to reference this in JavaScript outside of the library.

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.

Are You Using This Library?

If you're using this library I'd like to hear from you as I'm looking to provide some real life examples of it's use on day-to-day websites. Included is the chance to get a one-way link to your site (yes, for free!). Email me at info@biostall.com if you have an example I can look at. Thanks.