



Man muss schon gute Karten haben ...

Die Google Maps Erweiterung für Contao extern befüllt.

Wer steht denn da vorne?

- Christian de la Haye
- 46, aus Viersen
- im/mit dem Web selbständig seit 1996
- Autodidakt



Ziel für heute:

- Map mit Adresseneinträgen via Metamodels
- Eine einfache Markierung pro Datenbankeintrag
- Mit Info-Blase ...
- ... die sich auf Klick öffnet

Grundsätzliches zur Erweiterung

- Definition der Karte
- Erzeugung der Karten-Elemente
- Einbindung als CE/Modul

Aufbau im Frontend

- Bei CE und Modul gleich
- Einige Angaben im CE/Modul überschreibbar
 - z.B. Anzeigegröße,
 - Zoom-Faktor etc.

Grund: Je nach Befüllung wird nur 1 Karte für X verschiedene Ansichten benötigt

Grundsätzliches Template im FE

```
<div class= ... /div>
```

- **Html-Block mit**
 - Abmaßen, CSS-Klassen etc.

```
var gmap1_options = ...
```

```
... streetViewControl:true ...
```

```
... new google.maps.Marker ...
```

```
gmap1_dynmap(gmap1);
```

- **Javascript-Block mit**
 - Definierten Karten-Eigenschaften
 - Veröffentlichten Karten-Elementen
 - **Plus Aufruf einer nachgelagerten Funktion**

Ein Beispiel. Erst mal der Grundaufbau:

- Karte im Bereich „Googlemaps“ anlegen
 - ID der Karte notieren → Funktionsname
 - Markierungen: noch keine benötigt
- CE anlegen (oder Modul)
 - Die Seite sollte jetzt die „nackte“ Karte zeigen

Wechsel zum Testsystem

Das sehen wir uns im Workshop dann mal in einem Testsystem an.

Ist aber Contao-Standard und reine Bedienung der dlh_googlemaps.



Die Funktion gmap1_dynmap()

- Wird pro Karte generiert
 - (gmap1_dynmap, gmap2_dynmap...)
- Kann die komplette Karte verändern
 - Markierungen
 - Zentrierung
 - Zoom etc.

Die Funktion `gmap1_dynmap()`

- Kann an komplett anderer Stelle in der Seite stehen
 - Auch vor der eigentlichen Karte
- Kann durch andere Module eingefügt werden
 - „Eigenes Html“ reicht theoretisch auch, nur wegen dem `<script>`-Tag nicht unproblematisch.

Was muss/kann eingefügt werden?

- Die genannte Javascript-Funktion, darin:
 - Für jedes Kartenelement 1 Abschnitt
 - Für jeden Event-Listener 1 Abschnitt
- Wichtig: Alle Elemente gehören in die 1 Funktion pro Map.

Schema für eine einfache Markierung:

```
var gmap1_0_marker = new google.maps.Marker({  
    position: new google.maps.LatLng(51.24942, 6.39273),  
    map: gmap1,  
    title: "Irgendein Titel für den Marker",  
    zIndex: 1  
});
```

- Koordinaten
- ID der Map
- Titel

Schema für eine Infoblase:

```
var gmap1_0_infowindow = new google.maps.InfoWindow({  
  position: new google.maps.LatLng(51.24942, 6.39273),  
  pixelOffset: new google.maps.Size(-10, -10),  
  content: '<div style="width:200px;height:200px;">  
    Irgendein Inhalt für die Infoblase</div>'  
});
```

- Koordinaten
- Versatz der Infoblase zum Marker (Pixel)
- Inhalt – kann auch formatiert sein.

Schema für die klickbare Markierung:

```
google.maps.event.addListener(  
  gmap1_0_marker, 'click', function() {  
    gmap1_0_infowindow.open(gmap1);  
  });
```

- Name des Markers
- Name der Infoblase
- ID der Map

Woher nehme ich den Quelltext für andere Arten von Elementen?

Einfach aus den normalen Elementen der Erweiterung ein Code-Beispiel erstellen.

- Beispiel-Map anlegen
- Element anlegen
- Kartenelement erzeugen
- Passenden Quelltext rauskopieren
- Verallgemeinern (Platzhalter)

Jetzt wird's konkret: Metamodels

- Adressen werden in Metamodels verwaltet
- Koordinaten müssen schon vorhanden sein
 - (können auch via eigener Callback-Funktion gesetzt werden, aber das ist Metamodels-Sache)
- Eigener Karten-Marker pro Adresse

Metamodels – Step 1:

- Metamodel anlegen
- Eigenes Rendersetting für die Kartenausgabe
- Listenausgabe auf der Kartenseite anlegen

Wechsel zum Testsystem

Im Workshop geht es dann wieder zurück ins Testsystem – ein Standard-Metamodel integrieren.



Metamodels – Step 2:

- MM-Template „metamodel_kartenansicht“ erstellen
- MM-Template dem Rendersetting zuordnen
- Fertig.
Nach Belieben Filter erstellen, als Liste nochmal darunter ausgeben lassen, etc., etc.

Metamodels – das Template:

```
<?php $strRendersettings = ($this->settings)? 'settings' : 'view'; ?>
```

```
<?php if ($this->items->getCount()): ?>
```

```
<script>
```

```
function gmap1_dynmap(gmap1) {
```

Metamodels – das Template:

```
<?php
$count=0;

foreach($this->items->parseAll($this->getFormat(), $this->view) as
$arrItem):

    $koordinaten = $arrItem['text']['koordinaten'];

    $titel        = $arrItem['text']['firma'];

    $infoblase    = '<div style="width:200px;height:150px;">
<div>' . $arrItem['text']['firma'] . '</div>
<div>' . $arrItem['text']['strasse'] . '</div>
<div>' . trim($arrItem['text']['plz'] . ' ' . $arrItem['text']['ort']) . '</div>
<div>' . $arrItem[$this->getFormat()]['land'] . '</div>
</div>';

?>
```

Metamodels – das Template:

```
var gmap1_<?php echo $count; ?>_marker = new google.maps.Marker({
    position: new google.maps.LatLng(<?php echo $koordinaten; ?>),
    map: gmap1,
    title:"<?php echo $titel; ?>",
    zIndex: 1
});

var gmap1_<?php echo $count; ?>_infowindow = new google.maps.InfoWindow({
    position: new google.maps.LatLng(<?php echo $koordinaten; ?>),
    pixelOffset: new google.maps.Size(-10,-10),
    content: '<?php echo preg_replace('%(\r\n)|(\r)|(\n)%', '',
$infoblase); ?>'
});
```

Metamodels – das Template:

```
google.maps.event.addListener(  
gmap1_<?php echo $count; ?>_marker, 'click', function() {  
    gmap1_<?php echo $count; ?>_infowindow.open(gmap1);  
});  
  
<?php $count++; endforeach; ?>  
  
}  
</script>  
<?php endif; ?>
```

Wechsel zum Testsystem

Im Testsystem geht's jetzt in den Endspurt. Im Metamodel heisst es „raus aus der Liste – rein in die Karte“.

???



Geht noch mehr? Klar, z.B.:

- Automatische Clusterung der Markierungen
 - Vermeidet „Gedrängel“
 - Benötigt aber u.a. anderes Metamodel-Template
- Filterung der Einträge auf verschiedene Wege
 - Automatisch durch Verwendung der Metamodels-Filter

Und was das Google-API noch hergibt.

Beispiel für mehr: *www.heidetrends.de/bezugsquellen.html*

The screenshot shows a web browser window displaying the website www.heidetrends.de/bezugsquellen.html. The page features a header with the logo "HEIDETRENDS SAISON FÜR FARBE" and a navigation menu. Below the header is a search bar and a filter dropdown. The main content area is divided into two columns. The left column contains a map of Europe with numbered markers (1-15) indicating various locations. The right column is titled "TRENDWELTEN" and contains three sections: "KLASSISCH" (Classical), "MODERN", and "INNOVATIV" (Innovative). Each section includes a small image and a brief description. At the bottom of the page, there is a section titled "BEZUGSQUELLEN" (Sources) with the text "ERSTES AACHENER GARTENCENTER BECKERT E.K."

- Clustering
- Filter
- Detailanzeige in Mediabox

Website Heidetrends by gia-online
<http://heidetrends.de>
<http://gia-online.de>

Beispiel für mehr: www.emdr-ch.org/therapeutinnen.html

The screenshot shows the website www.emdr-ch.org/therapeutinnen.html in a Firefox browser. The page title is "Zertifizierte EMDR-TherapeutInnen". The main content is a map of Switzerland with blue circular markers indicating the locations of certified therapists. On the left side, there are several filter options:

- PLZ-Bereich:** A search box for postal codes.
- Kanton:** A dropdown menu set to "Nicht filtern".
- Anrede:** Radio buttons for "Frau" and "Herr".
- Sprachen:** A list of languages with checkboxes: Bosnisch, Deutsch, Englisch, Französisch, Griechisch, Italienisch, Niederländisch, Norwegisch, Portugiesisch, Rumänisch, Russisch, Schwedisch, Serbo-Kroatisch, Spanisch, Thai, and Türkisch.

Below the map, a list of therapists is displayed:

- Aebischer, Lucia, lic.phil., 1700 Fribourg
- Aeschlimann-Sallin, Magalie, 1752 Villars-sur-Glâne
- Agoc, Elisabeth, Dr. med., 9004 St. Gallen

- Clustering
- Filter
- Tags

Website EMDR Schweiz by DTP Atelier
<http://www.emdr-ch.org>
<http://dtp-atelier.com>

Auflistungsmodul, eigene Scripte

- Prinzipiell gleicher Aufbau
- Der Karte ist es egal, wer oder was die Einträge erzeugt
- Ausgabememplate muss „nur“ das entsprechende Javascript ausgeben

Auflistung – fast identisches Template:

```
<?php
$count=0;

foreach($this->tbody as $arrItem):

    $koordinaten = $arrItem['koordinaten']['content'];
    $titel        = $arrItem['firma']['content'];
    $infoblase    = '<div style="width:200px;height:150px;">
<div>' . $arrItem['firma']['content'] . '</div>
<div>' . $arrItem['strasse']['content'] . '</div>
<div>' . trim($arrItem['plz']['content'].' '.$arrItem['ort']
['content']) . '</div>
<div>' . $arrItem['land']['content'] . '</div>
</div>';

?>
```

Wechsel zum Testsystem

Und auch die Ausgabe über das Contao-Modul „Auflistung“ sollte unser Testsystem jetzt können.



Infos und Weitergehendes

- **Google-API**

- <https://developers.google.com/maps/documentation/javascript>

- **Clustering**

- <https://developers.google.com/maps/articles/toomanymarkers?hl=de>

- **Handbuch (leider für ältere Version)**

- http://www.delahaye.de/dlh_googlemaps.html

- **Contao Forum**

- https://community.contao.org/de/forumdisplay.php?122-dlh_googlemaps

Vielen Dank für die Aufmerksamkeit!



Christian de la Haye
Karl-Seepe-Str. 12
41747 Viersen

02162 501225
service@delahaye.de



Contao
KONFERENZ