CVO Antwerpen

Project Oman

Persoonlijk dossier

Oualid Yousfi
25-5-2015

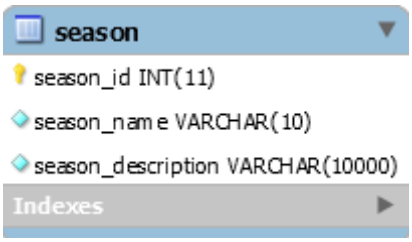
Inhoud

Inleiding

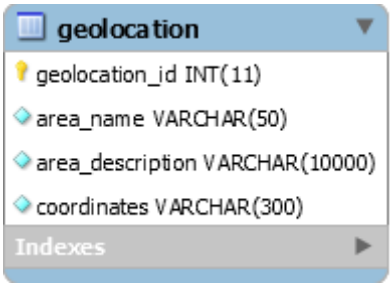
Eerst zal ik kort de database bespreken aangezien Tom en ik de basis voor de database hadden aangelegd, vervolgens de DAL klassen waar ik aan gewerkt heb en de service-side validatie. Tenslotte zal ik eindigen met de Google Maps implementatie.

Database

De eerste belangrijke stap dat we moesten bespreken is hoe we *de mogelijkheid van twee talen kunnen incorporeren in de database*. Daarmee maken we gebruik van extra tabellen. Een voorbeeld hiervan is de tabel **Season**. Je kan er een enumeratie veld van maken aangezien het aantal seizoenen beperkt is maar zo zit je dan vast met seizoenen in één specifieke taal. Als je ze in een ander apart tabel steekt kan je zoveel seizoenen toevoegen in zoveel mogelijk talen als je wilt.

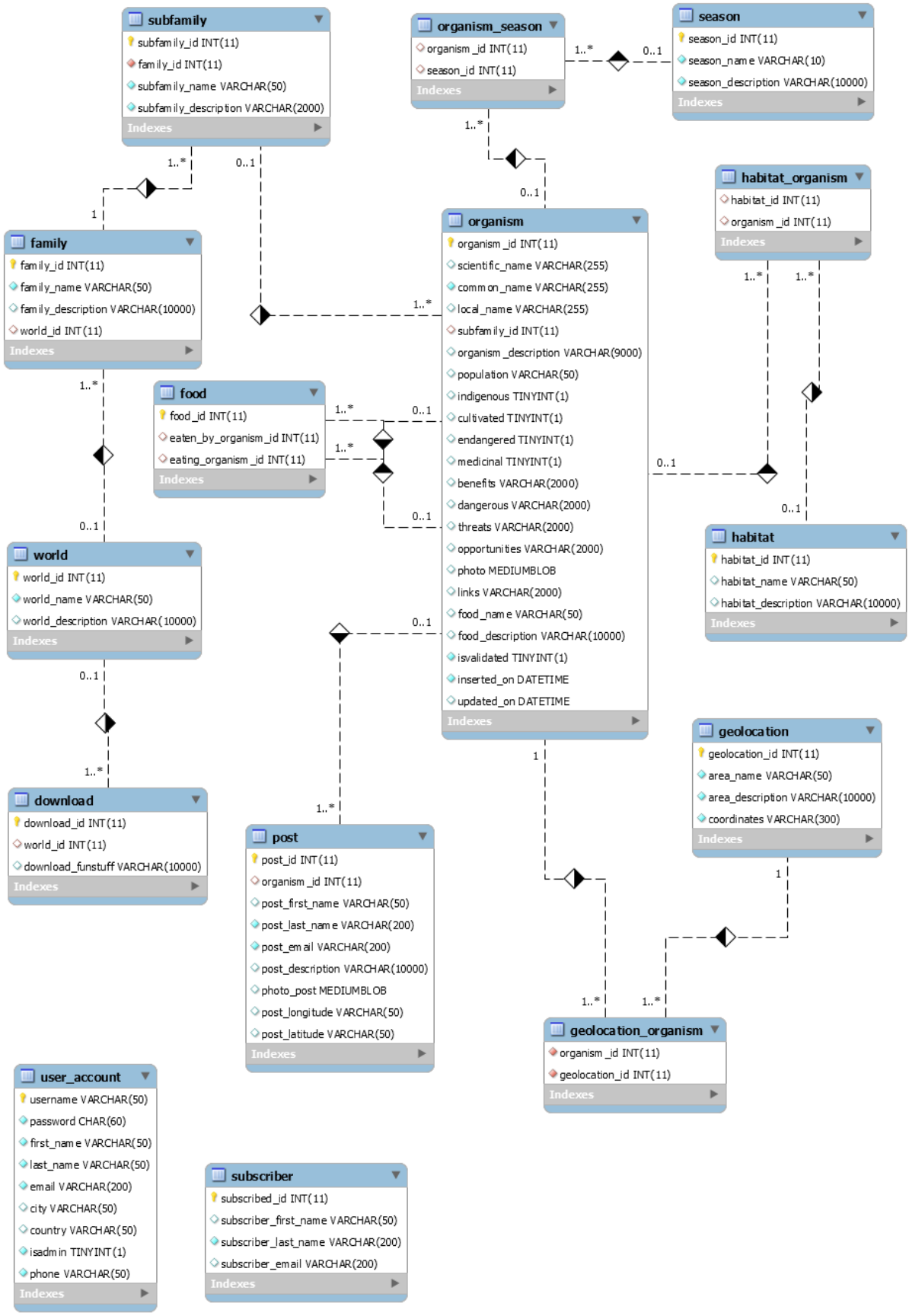


Initieel had **Geolocation** veel meer velden vrijgemaakt om de coördinaten op te slaan, zoals latitude coördinaat 1 en longitude coördinaat 1. Daarom heb ik gesuggereerd om de coördinaten in één veld genaamd Coordinates waar je alle coördinaten kan insteken en elk coördinaat kan onderscheiden aan de hand van een separator (in dit geval een eenvoudige komma).



Onze database is gecentraliseerd rond de entiteit **Organism**. Deze entiteit heeft relaties (direct of indirect) met bijna elk ander tabel. Het volledige diagram heb ik toegevoegd zodat dit duidelijk wordt.

Volledige database diagram



DAL

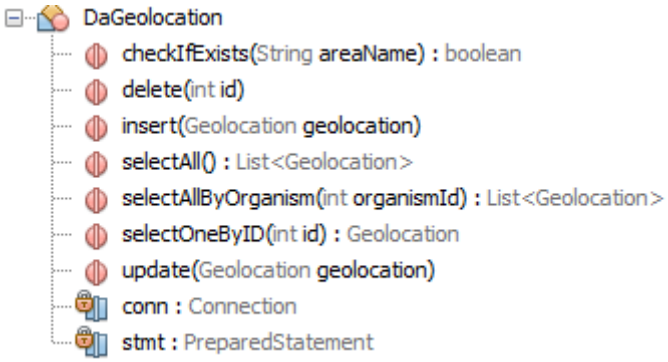
De DAL klassen **DaDownload**, **DaGeolocation**, **DaSubfamily** werden door mij gemaakt. Elke DAL klasse heeft standaard CRUD methodes. De queries werden correct geparametriseerd.

```
stmt = conn.prepareStatement("UPDATE geolocation SET area_name=?, "
                + "area_description=?, coordinates=? WHERE geolocation_id=?");
stmt.setString(1, geolocation.getAreaName());
stmt.setString(2, geolocation.getAreaDescription());
stmt.setString(3, geolocation.getCoordinates());
stmt.setInt(4, geolocation.getGeolocationId());
```

Er werd gebruik gemaakt van de **DataSource** klasse om een verbinding met de database te maken.

```
conn = DataSource.getConnection();
```

Dit is hoe elke DAL klasse er uit ziet. Ieder van hen heeft ook een connection en een private statement variabele die hergebruikt word voor elke DAL methode. Zoals eerder vermeld werd heeft elke klasse een insert, selects, update en een delete. checkIfExists is om te controleren of het al bestaat in de database.



Zelf heb ik ook geïmplementeerd dat wanneer je een Geolocation verwijdert of elk ander entiteit dat ook gebruik maakt van een tussentabel dat dan de gerelateerde records in de tussentabellen ook worden verwijderd met behulp van een cascade.

Service-side validatie

Om de gegevens te controleren heb ik validatie toegevoegd in de service-laag. Met behulp van regex controleer ik of de gegevens toelaatbaar zijn. Hiervoor zijn verscheidene regex strings klaargemaakt, eentje voor input-velden, eentje voor tekst-velden (\r\n's toegelaten), eentje voor telefoonnummers, eentje voor e-mails enzovoort.

```
if (!name.matches("[^()[¥¥¥¥]{}*&^%$<>#0-9@!]+$")) {
    return "required";
} else if (!description.matches("^[^<>¥¥¥¥/{}¥¥[¥¥]]*(¥¥¥r¥¥¥n)?$")){
    return "required";
} else if (coordinates.length() < 110) {
    return "map";
} else if (DaGeolocation.checkIfExists(name) == false) {
    Geolocation geolocation = new Geolocation();
    geolocation.setAreaName(name);
    geolocation.setAreaDescription(description);
    geolocation.setCoordinates(coordinates);
    DaGeolocation.insert(geolocation);
    return "succes";
} else {
    return "exists";
}
```

Voor **Geolocation** is er een speciale validatie door mij toegevoegd. Als er minder dan drie coördinaten worden doorgestuurd (lengte van string korter dan 110 karakters) dan worden de gegevens niet geaccepteerd en krijg je een melding om een map eerst te maken.

Zoals je kan zien zijn er verscheidene unieke boodschapmeldingen naargelang welke validatie uitgevoerd wordt. Als het om de name of description gaat krijg je een foutmelding terug van dat die velden required zijn. Als je niet genoeg coordinaten doorgeeft krijg je terug dat je eerst een map moet aanmaken. Als er al een geolocatie met dezelfde naam in de database zit krijg je een foutmelding dat die al bestaat. Wanneer je door alle validatie geraakt krijg je een melding dat de actie is gelukt.

Deze validatie is niet alleen aanwezig in alle inserts maar ook alle updates.

Google Maps

Nu zal ik de Google Maps implementatie deel per deel bespreken. Allereerst heb ik de javascript in één file gestoken zodat het gescheiden was van de andere javascript. Daarna heb ik ervoor gezorgd dat globale variabelen werden gelimiteerd tot één array, een soort van mini-config voor de hele Google Maps script.

```
var settings = {area: [], areaPath: null, canvas: 'map-canvas', map: null,
    place: 'area-coordinates', mapOptions: {center: {lat: 20.8723303,
            lng: 55.765891}, zoom: 7, minZoom: 6, maxZoom: 11,
        mapTypeControl: false, streetViewControl: false,
        disableDoubleClickZoom: false}};
```

De belangrijkst functie in dit script is de **initialize()** functie die de map creëert wanneer het wordt aangeroepen. Maar dit is niet de eerste functie die wordt aangeroepen.

De script referentie naar de Google Maps API is niet initieel aanwezig op de HTML pagina. Die wordt dynamisch toegevoegd wanneer de map wordt aangeroepen. Dit gebeurt aan de hand van **loadScript().**

```
function loadScript() {
    var check = document.getElementById('maps-script');
    if (check === null) {
        var script = document.createElement('script');
        script.id = 'maps-script';
        script.type = 'text/javascript';
        script.src = 'https://maps.googleapis.com/maps/api/js?v=3.19exp' +
                '&callback=initialize&language=en';

        document.body.appendChild(script);
    } else {
        initialize();
    }
}
```

Eerst kijkt men of er een DOM element met de ID ‘maps-script’ bestaat. Als die nog niet aanwezig is wordt die gecreëerd, zoniet dan wordt de initialize() functie aangeroepen. Waarom terug de initialize() functie aanroepen als de map al bestaat? Het dients als een refresh voor de map, als je iets insert dan blijft het gebied die je hebt aangeduid op de map nog staan. Door ze nog eens keer te initialiseren verwijder je mogelijke aanpassingen die een gebruiker voor je heeft gemaakt en kan je beginnen met een nieuwe map.

Misschien zit je af te vragen dat wanneer er een nieuwe script referentie wordt toegevoegd hoe dat er voor zorgt dat de map eigenlijk wordt gecreëerd. Je kunt er namelijk voor kiezen om in de API URL een functie toe te voegen die asynchronisch wordt uitgevoerd (‘&callback=initialize’). Die roept de initialize() functie op en die wordt aangeroepen.

initialize() functie

```
function initialize() {
    settings.map = new google.maps.Map(document.getElementById(settings.canvas),
            settings.mapOptions);

    settings.map.set('styles', [{featureType: "all", stylers: [
                {saturation: 60}]}]);

    google.maps.event.addListener(settings.map, 'click', function (event) {
        makeArea(event.latLng);
    });

    settings.areaPath = new google.maps.Polygon({
        strokeColor: '#FF0000',
        strokeOpacity: 0.4,
        strokeWeight: 3,
        fillColor: '#FF0000',
        fillOpacity: 0.35,
        editable: true
    });

    google.maps.event.addListener(settings.map, 'mouseout', function () {
        showCoordinates(settings.areaPath, settings.place);
    });

    showArea();

    function makeArea(location) {
        settings.area.push(location);
        showArea();
    }
}
```

Hier wordt de map gecreëerd met al zijn settings. Er zijn ook een aantal listeners toegevoegd aan de map, eentje voor alle muiskliks die de functie **makeArea()** aanroept. En een tweede listener voor elke keer als je met de muis beweegt van de map naar de submit knop dat er de **showCoordinates()** functie wordt aangeroepen. Daarnaast wordt er altijd de functie **showArea()** eenmalig aangeroepen met elke initialize() call die het aangeduide gebied toont op de map. Als het een nieuwe map is doet dit niets maar met een update wordt de bestaande map aangeroepen. De makeArea() functie stuurt bij elke muisklik de coördinaat door en roept de functie showArea() op. Die is relatief simpel, als er meer dan twee coördinaten zijn wordt het gebied getoond (met drie coördinaten kan je een driehoek maken).

```
function showArea() {
    if (settings.area.length > 2) {
        settings.areaPath.setPath(settings.area);
        settings.areaPath.setMap(settings.map);
        settings.area = [];
    }
}
```

De array die de coördinaten bijhoudt wordt ook altijd gereset dus als je op de map een gebied hebt kan je nog eens keer drie keer klikken en dan heb je een nieuw gebied en de oude is verdwenen.

Dan showCoordinates() zorgt er gewoon voor dat de coördinaten in een verborgen veld worden gestoken en correct geformatteerd.

```
function showCoordinates(areaPath, placecoord) {
    var string = '';

    var coordinates = areaPath.getPath();
    for (var i = 0; i < coordinates.getLength(); i++) {
        var xy = coordinates.getAt(i);
        string += "(" + xy.lat() + ',' + xy.lng() + ')@';
    }

    var coord = document.getElementById(placecoord);
    coord.value = string.slice(0, -1);
}
```

Dat was het eerste deel van het script. Nu het tweede. Er zit ook een beetje JQuery in dit script. Dit bepaalt wat er gebeurt wanneer. Wanneer je op de edit knop klikt in update, wanneer je op de reset knop klikt, wanneer je de pop-up venster sluit.

```
$(document).ready(function () {
    $('#edit-map').click(function () {
        $('#edit-map').hide();
        $('#update-canvas').show();
        settings.canvas = 'update-canvas';
        settings.place = 'update-coordinates';
        getCoordinates();
        loadScript();
    });

    $('#geolocation-close1, #geolocation-close2').click(function () {
        settings.canvas = 'map-canvas';
        settings.place = 'area-coordinates';
        settings.mapOptions.center = {lat: 20.8723303, lng: 55.765891};
        resetArea();
    });

    $('#geolocation-reset1, #geolocation-reset2').click(function () {
        resetArea();
    });
});
```

Hier komen twee nieuwe functies **getCoordinates()** en **resetArea()** tevoorschijn. Je ziet ook dat er twee canvas-IDs en coördinaten-IDs zijn. Default staat het op ‘map-canvas’ en ‘area-coordinates’. Het is alleen wanneer je gaat updaten dat het wordt gewijzigd en teruggezet wanneer je klaar bent.

getCoordinates() zorgt er voor dat de coördinaten die in het verborgen veld steken correct worden geformatteerd zodat je er een map van kan maken. Niet alleen dat maar het controleert welke coördinaten het zijn en zorgt ervoor dat het gebied midden in de map wordt gezet. Dit wordt alleen gebruikt bij een update.

```
function getCoordinates() {
    var input = document.getElementById('update-coordinates').value;
    var split = input.split("@");
    var maxLat = 0.0;
    var minLat = 500.0;
    var maxLng = 0.0;
    var minLng = 500.0;
    if (input.length > 0) {
        for (var i = 0; i < split.length; i++) {
            var part = split[i].split(",");
            var latitude = part[0].slice(1, part[0].length);
            maxLat = ((maxLat > parseFloat(latitude)) ? maxLat : parseFloat(latitude));
            minLat = ((minLat < parseFloat(latitude)) ? minLat : parseFloat(latitude));
            var longitude = part[1].slice(0, -1);
            maxLng = ((maxLng > parseFloat(longitude)) ? maxLng : parseFloat(longitude));
            minLng = ((minLng < parseFloat(longitude)) ? minLng : parseFloat(longitude));
            settings.area.push({lat: parseFloat(latitude), lng: parseFloat(longitude)});
        }
        settings.mapOptions.center = {lat: (maxLat + minLat) / 2, lng: (maxLng + minLng) / 2};
    }
}
```

resetArea() doet een reset en verwijdert het gebied van de map.

```
function resetArea() {
    settings.areaPath.setMap(null);
    settings.area = [];
    settings.areaPath = new google.maps.Polygon({strokeColor: '#FF0000',
        strokeOpacity: 0.4, strokeWeight: 3, fillColor: '#FF0000',
        fillOpacity: 0.35, editable: true});
    var coord = document.getElementById(settings.place);
    coord.value = "";
}
```

Google Maps problemen

- Het laden van de map zorgde voor wat problemen. Je kan de map niet direct laden wanneer de site laadt want de map wordt opgeroepen in een pop-up. Op die manier wordt de map niet goed opgeroepen en krijg je overal grijze gebieden op de map. Bij de insert heb ik dit opgelost door loadScript() te laden wanneer er op de knop die de insert pop-up tevoorschijn haalt klikt, een onclick event dus. Bij update werkt dit niet aangezien er gegevens worden opgehaald dus heb ik een knop op de update pop-up gezet die de map aanroept wanneer je erop klikt.
- Als je wilt dat javascript iets expliciet als een integer/float wilt behandelen moet je parseFloat gebruiken. Eerst had ik dit niet gedaan en omdat ik de string methode split had gebruikt werden latitude en longitude als string beschouwd.