Mathieu De Smet

Fietsen met ASP.NET Ajax

GEBRUIK MICROSOFT LOCAL LIVE OP JE WEBSTEK

Routeplanners zijn niet meer weg te denken uit ons dagelijks bestaan. Ze helpen je snel van punt A naar B, of het nu met de auto is of te voet, of je nu de kortste weg wilt of de snelste. Je weet meteen hoe ver je moet rijden en wat je route is. Maar wat als je eens een fietstochtje wilt maken? Dan moet je aan de slag met kaarten.

In België heb je het knooppuntennetwerk van de provincies. Limburg is hiermee gestart. Op verschillende plaatsen vind je een aanduiding langs de weg met een nummer en de verbindingen naar de volgende nummers. Je koopt een kaart en stippelt de route uit van knooppunt naar knooppunt. Een fietstocht maken is dan eenvoudig, gewoon een lijst van knooppunten afwerken. Maar op die manier is het wel moeilijk te berekenen hoe ver je gaat fietsen. En wil je in een andere regio fietsen, dan moet je de juiste kaart hebben. Het zou gemakkelijker zijn als je een routeplanner zou kunnen gebruiken, maar die gaat enkel van punt A naar punt B. Dus het is of langs dezelfde weg terugkeren, of de trein terug nemen. Tijd dus voor een nieuw initiatief, de routeplanner voor fietsers die je meteen thuisbrengt.Wat hebben we nodig om een routeplanner op de computer te maken?

- Een lijst van de verschillende knooppunten. Per knooppunt een omschrijving en een plaatsaanduiding op de kaart.
- Een lijst van de verbindingen tussen deze knooppunten. Van elke straat willen we ook weten hoe lang ze is.
- Een programma waarmee we de knooppunten op een kaart kunnen visualiseren.
- Een manier om de berekende route op onze GPS te plaatsen, zodat we tijdens de fietstocht geen pc met internetverbinding nodig hebben.

Een Mashup tussen Windows Local Live en Microsoft ASP.NET Ajax leek mij een goede oplossing om een mooie, maar toch eenvoudige, user-interface te maken.

Windows Live Local op je internetpagina

Met Windows Live Local kun je al snel een kaart op je eigen website tevoorschijn halen en er op de gewenste plaats Push-Pins op plaatsen. Windows Live Local bestaat uit een hele set instructies, verwerkt in een javascript-file, die je op jouw website inlaadt. Windows Local Live werkt zowel op Microsoft Internet Explorer, Firefox als andere browsers. Alle scripting kun je dan ook het beste in javascript schrijven om volledig compatibel te blijven met deze browsers. De kaart wordt ingeladen in een DIV-tag die je op jouw website plaatst. Op die manier bepaal je zelf waar de kaart komt en hoe groot ze moet worden weergegeven. Zet je via de style de hoogte en breedte op 100%, dan heb je een kaart in full-screen mode. Je kunt de kaart ook verkleinen, zodat hij in jouw website past. De kaart moet worden ingeladen en geïnitialiseerd bij het openen van de webpagina. De beste manier om ze te initialiseren is via een script bij het OnLoad-event van de BODY-tag. Op de pagina moet een globale variabele worden aangemaakt die later kan worden gebruikt bij het manipuleren van de kaart. Met de instructies van codevoorbeeld 1 heb je al snel een volledig werkende kaart op jouw internetpagina. Zoals bij elk kaartsysteem wordt een unieke plaats op de kaart bepaald door de Longitude- en Latitude-coördinaten.

Er bestaan allerlei programma's om andere systemen zoals UTM of Lambert om te zetten naar een Lon/Lat-combinatie. Van de provincie Vlaams-Brabant kreeg ik de coördinaten van de knooppunten van de fietspaden in een Microsoft Excel-bestand in Lamber 87-formaat.

FietsRoute Library

Om de routeplanner te maken, hebben we een business-objectlayer nodig die uit de volgende functionaliteiten bestaat:

- Een lijst van knooppunten
- De verbindingen tussen de knooppunten
- En een programma waarmee je een fietstocht kan berekenen.

In dit artikel ga ik niet verder in op de inhoud van deze classen. De FietsRoutePoints-class geeft ons een overzicht van alle mogelijke fietsrouteknooppunten. Voor elk knooppunt hebben we volgende informatie: een code, een omschrijving en de geografische coördinaten. In de FietsRouteRoads-class vinden we alle mogelijke verbindingen tussen twee fietsrouteknooppunten. Voor elke verbinding hebben we volgende informatie: het beginpunt, het eindpunt en de echte afstand tussen deze twee punten. De FietsRouteCalculation-class zorgt ervoor dat de juiste route wordt berekend aan de hand van het beginpunt, het eindpunt en de afstand die je wenst af te leggen. Het resultaat van deze berekening is een lijst van verbindingen tussen telkens twee knooppunten.

De user-interface

Om een fietsroute te berekenen hebben we drie belangrijke zaken nodig:

- Een startpunt, dat ook ons eindpunt is
- De afstand die we willen fietsen
- Een lijst van de berekende routes.

Ik heb er voor gekozen om bij een eenvoudige user-interface steeds hetzelfde begin- en eindpunt te nemen. Veel fietsers rijden immers met hun fiets in de auto naar een mooie plek, om van

```
<script src="http://dev.virtualearth.net/mapcontrol/v3/mapcontrol.js">
</script>
<body onload="initMap();">
<script language="javascript">
var map = null;
function initMap()  {
   if ( map == null ) {
      map = new VEMap('myMap');
      map.LoadMap();
   }
}
</script>
<div id="myMap" style="width: 100%; height: 100%; left: 0px; top: 0px;" />
```

Codevoorbeeld 1.

```
<asp:ListBox ID="ListBoxMaxDistance" runat="server" AutoPostBack="true"
 Rows="1">
   <asp:ListItem Selected="true" Value="40" Text="40"></asp:ListItem>
   <asp:ListItem Value="60" Text="60"></asp:ListItem>
   <asp:ListItem Value="80" Text="80"></asp:ListItem>
   <asp:ListItem Value="100" Text="100"></asp:ListItem>
</asp:ListBox>
<asp:ListBox ID="TextBoxStartPoint" runat="server" DataSourceID=
     "DataSourcePoints" AutoPostBack="true" DataTextField="name"
     DataValueField="name" Rows="1"/>
<asp:ObjectDataSource ID="DataSourcePoints" runat="server"
 SelectMethod"toDataTable" TypeName=" FietsRoute4.FietsRoutePoints" />
```

Codevoorbeeld 2.

daaruit een fietstocht te maken. De gebruiker zal dus uit een van de knooppunten kiezen en een fietsafstand bepalen. De bedoeling is dat we telkens een van beide gegevens aanpassen en zo een lijst met mogelijke routes krijgen waaruit de gebruiker een keuze kan maken. We willen dit doen zonder dat de gebruiker eerst op een knop 'Bereken route' moet drukken.

De eerste control, een ListBox-control, plaatsen we voor de keuze van de fietsafstand. Om het de gebruiker gemakkelijk te maken, voorzien we in vier standaard afstanden: 40, 60, 80 en 100. De lijst met mogelijke fietsknooppunten waar de fietser kan starten, halen we uit het business-class-object FietsRoutePoints. In deze class heb ik in een method (functie) voorzien die de lijst van fietsknooppunten weergeeft als datalist. We maken een ObjectDataSource-control aan en koppelen deze aan de method uit de fietsRoutePoints-class. De ListBox-control om de startpunten weer te geven, koppelen we met een DataSourceID aan de ObjectDataSource. In codevoorbeeld 2 kun je de opbouw voor beide ListBox-controls vinden. Voor beide ListBox-controls zetten we de property AutoPostBack op True. Op die manier weet ASP.NET dat elke wijziging een postback tot gevolg heeft en hiervan kunnen we gebruikmaken om een berekening uit te voeren. Als derde en laatste control, maken we gebruik van een GridView-control. In deze control geven we alle mogelijke berekende routes weer. We plaatsen een tweede ObjectDataSource op de pagina die verwijst naar de FietsRoute-Calculation-method (functie) uit het businessobject. Voor deze object-datasource hebben we twee parameters als input nodig: de afstand en het beginpunt. Het resultaat van deze functie na berekening is een dataset met daarin de afstand van elke berekende route. De Gridview bestaat uit twee kolommen, een hyperlink om de gekozen route op het scherm te plaatsen en een Label om de reële afstand te tonen. De implementatie van de GridView-controle vind je in codevoorbeeld 3.

```
<asp:GridView ID="GridView1" runat="server"
 DataSourceID="DataSourceCalc ulation" AutoGenerateColumns="False" >
<Columns>
   <asp:TemplateField>
      <ItemTemplate>
      <%#"<a href=""javascript:include('route.aspx?r=" & Eval("route") &
      "');""><img src=Images/preview.jpg></a>"%>
      </ItemTemplate>
   </asp:TemplateField>
   <asp:BoundField DataField="km" />
</Columns>
</asp:GridView>

<asp:ObjectDataSource ID="DataSourceCalculation" runat="server"
SelectMethod="Calculate" TypeName="FietsRoute4.FietsRouteCalculate">
   <SelectParameters>
      <asp:FormParameter FormField="TextBoxStartPoint" Name="StartPoint"
                         Type="String" />
      <asp:FormParameter FormField="ListBoxMaxDistance" Name="Distance"
                         Type="Int32" />
   </SelectParameters>
</asp:ObjectDataSource>
```

Codevoorbeeld 3.

ASP.NET AJAX

Na het kiezen van een startpunt en de afstand zouden we vroeger in een knop 'Start berekening' voorzien op de html-pagina. Deze knop zou een PostBack naar de server genereren waar de route berekend wordt en een nieuwe pagina wordt opgemaakt met de gekozen route. Het nadeel is dat de volledige pagina opnieuw naar de gebruiker wordt gestuurd, terwijl maar een klein deel moet worden geüpdatet. De ASP.NET Ajax-technologie laat toe om delen van een internetpagina te updaten met nieuwe informatie van de server, zonder hiervoor de volledige pagina te laden. Dit geeft een veel mooiere user-interface en een snellere laadtijd. Microsoft heeft een Framework-library gemaakt, ASP.NET ATLAS, waarmee de Ajax-technologie eenvoudig toegankelijk is binnen ASP.NET-pagina's. In de plaats van een knop 'Start berekening' te maken, kunnen we er via ATLAS voor zorgen dat elke wijziging in het startpunt of in de afstand automatisch een berekening tot gevolg heeft en dus ook een nieuwe lijst met resultaten toont in de Grid-View-control. Dit alles asynchroon, zodat de user-interface niet wordt geblokkeerd tijdens de berekening. We gaan deze functionaliteit toevoegen aan de reeds bestaande controls. Na de installatie van ASP.NET ATLAS krijg je een reeks nieuwe website-templates in Microsoft Visual Studio. Je kiest File➞New Website en gebruikt de 'ASP.NET AJAX-enabled' website-templates om een lege website te creëren. Je kunt hiervoor kiezen uit een website met je favoriete programmeertaal: Visual Basic, C#, of J#. De scriptmanager-tag (<asp:ScriptManager ID="ScriptManager1" runat="server" />) is de eerste tag die reeds op de standaardpagina komt. Elke pagina waar je de functionaliteit van ASP.NET AJAX wilt gebruiken, moet zo'n tag bevatten. Om bepaalde delen van een website te kunnen updaten met een roundtrip naar de server, zonder dat hiervoor de volledige webpagina opnieuw moet worden geladen, gebruiken we een UpdatePanel-tag. De bestaande GridView-control verplaatsen we naar de ContentTemplate-tag van de UpdatePanel-control. Op die manier kan de GridView worden aangepast met een asynchrone call naar de server zonder de volledige webpagina opnieuw in te laden. De refresh van de UpdatePanel-control kan worden geactiveerd door middel van events uit andere controls op de HTML Form, zoals de SelectedIndexChanged-event uit de ListBox-control. In de Trigger-tag specificeren we een AsyncPost-BackTrigger met de ID van de control en naam van het event. ASP.NET ATLAS zorgt voor alle verwerking en nodige client-scriptcode om het geheel aan elkaar te laten passen. Dus telkens als de index in één van de ListBox-controls wijzigt, wordt de UpdatePanel via een asynchrone postback naar de server de GridView opnieuw opgemaakt en teruggeplaatst in de user-interface. Zie codevoorbeeld 4. ASP.NET ATLAS kent ook een mogelijkheid om tijdens het berekenen van de route en het opnieuw inladen van de GridView-control een boodschap op het scherm te tonen. Het is voldoende om een UpdateProgress-control toe te voegen op de html-pagina met de boodschap die moet worden getoond tijdens het wachten. Dit kan een tekst zijn, een image, of eender welke html-code. Zie codevoorbeeld 5 voor de implementatie ervan. In afbeelding 1 kun je het resultaat zien.

Het tonen van een route op de kaart

Nu het programma de mogelijke routes heeft berekend en weergeven in een GridView-control, moet de gebruiker deze route kunnen bekijken op de Windows Local Live-kaart. In de Windows Local Live-library vinden we een class VEPolyline. Deze class zorgt voor het tekenen van lijnen in een bepaalde kleur en dikte tussen verschillende punten. Het belangrijkste hierbij is dat ze een array hebben met alle GPS-coördinaten van de punten die we wensen te verbinden met een lijn. Aan de hand van de berekende route en de lijst van knooppunten in het businessobject kunnen we zo'n lijst samenstellen. Deze lijst moet worden verwerkt in een stuk javascript-code en ingeladen worden in de browser. Om dit te doen, heb ik volgende techniek gebruikt; je had misschien al gezien dat in de datagrid een <asp:TemplateField> werd gebruikt met een

```
<asp:ScriptManager runat="server" ID="scriptManager1" EnablePartial
     Rendering="true" />
<asp:UpdatePanel ID="OverzichtRoute" runat="server"
UpdateMode="Conditional" RenderMode="Inline">
   <Triggers>
      <asp:AsyncPostBackTrigger EventName="SelectedIndexChanged"
                                ControlI D="TextBoxStartPoint" />
      <asp:AsyncPostBackTrigger EventName="SelectedIndexChanged"
                                ControlI D="ListBoxMaxDistance" />
   </Triggers>
   <ContentTemplate>
      <asp:GridView ID="GridView1" runat="server"
                    DataSourceID="DataSourc eCalculation  AutoGenerateColu
                    mns="False" >
         <Columns>
            <asp:TemplateField>
               <ItemTemplate>
                    <%#"<a href=""javascript:include('route.aspx?r=" &
               Eval("route") & "');""><img src=Images/preview.jpg></a>"%>
               </ItemTemplate>
            </asp:TemplateField>
         <asp:BoundField DataField="km" />
      </Columns>
      </asp:GridView>
   </ContentTemplate>
</asp:UpdatePanel>
<asp:ObjectDataSource ID="DataSourceCalculation" runat="server"
                      SelectMethod="GetRoutes" TypeName="FietsRouteDataset">
   <SelectParameters>
      <asp:FormParameter FormField="TextBoxStartPoint" Name="StartPoint"
                         Type="String" />
      <asp:FormParameter FormField="ListBoxMaxDistance"
                         Name="MaxDistance" Type="Int32" />
   </SelectParameters>
</asp:ObjectDataSource>
```

Codevoorbeeld 4.

```
<asp:UpdateProgress runat="server" ID="myUpdateProgress"
                    DynamicLayout="true">
   <ProgressTemplate>
   Even geduld
   </ProgressTemplate>
</asp:UpdateProgress>
```

Codevoorbeeld 5.

```
<script src="http://dev.virtualearth.net/mapcontrol/v3/mapcontrol.js"></
script>
<body onload="initMap();">
<script language= "javascript">
var map = null;
function initMap()  {
   if ( map == null ) {
      map = new VEMap('myMap');
      map.LoadMap();
   }
}
</script>
<div id="myMap" style="width: 100%; height: 100%; left: 0px; top: 0px;" />
```

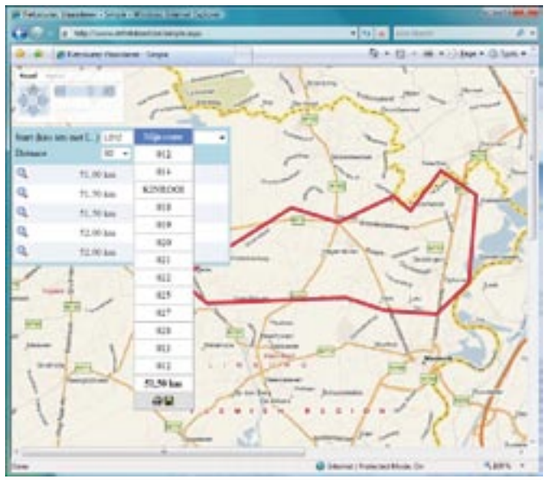
Codevoorbeeld 6.

Live library de verbindingslijntjes (polygon) op de kaart zal tekenen. De gekozen route wordt ingelezen vanuit de querystring en doorgegeven aan het businessobject om een lijst van de GPS-coördinaten te krijgen. De GPS-coördinaten worden in een javascript-array verwerkt. Deze array wordt gebruikt in de VEPolyline-class van Microsoft Local Live. Met de functie AddPolyline van de map-class worden lijntjes getoond op de kaart. Telkens als de gebruiker een route kiest, wordt een javascript-programma gegenereerd op de server en uitgevoerd op de client, in de browser. In afbeelding 1 zie je het eindresultaat. Op http://www.defietskaart.be/simple.aspx kun je het hele programma bekijken.

Tot slot

Met Microsoft Local Live is het eenvoudig om je eigen informatie op een kaart weer te geven. Veel codevoorbeelden vind je op http://dev.live.com/virtualearth/sdk/. Een Mashup tussen verschillende technologieën is niet alleen leuk om te doen, maar geeft je ook snel een goed resultaat. Er bestaan heel veel webservices die je kunt gebruiken om zelf jouw eigen combinatie te maken zonder dat je telkens alles opnieuw hoeft te bedenken. Belangrijk is wel dat je providers gebruikt waar een degelijke SDK of handleiding voorhanden is.

link naar een javascript-functie. Deze functie zorgt er voor dat een javascript-file op de server wordt ingeladen en uitgevoerd in de client. De bedoeling is nu dat het javascript-programma telkens de juiste route weergeeft. Hiervoor heb ik een ASPX-file gemaakt die dit stuk javascript genereert. De output van de ASPX-file zal dus een javascript zijn. In codevoorbeeld 6 zie je als eerste de javascript-functie die de ASPX-file in de browser voor verwerking zal oproepen en inlezen. De ASPX-pagina Route.apx zorgt voor het opmaken van het javascript-gedeelte dat met de Microsoft Local



Afbeelding 1. Het resultaat van de implementatie

Mathieu De Smet is zelfstandig .NET-ontwikkelaar. Hij ontwerpt en implementeert web-based administratieprogramma's voor Belgische pensioenfondsen. Hij is te bereiken op mds@mdssoft.net

Referenties

De handigste tips vind je op de interactieve SDK via de URL : http://dev.live.com/virtualearth/sdk/

De fietskaart in actie: http://www.defietskaart.be