Gill Cleeren is senior developer bij Ordina Belgium

Ajax zonder moeite

WEBAPPLICATIES IN ASP.NET AJAX

Webapplicaties verdringen steeds vaker desktopapplicaties. Om echter dezelfde user experience te krijgen, moet de interactiviteit verbeterd worden. Hiervoor is Ajax de perfecte oplossing. Met ASP.NET AJAX wordt het voor ASP.NET-ontwikkelaars een koud kunstje om de webapplicatie van morgen te bouwen. In dit artikel beschrijft de auteur kort de belangrijkste facetten van het nieuwe platform.

Webapplicaties gebruiken het http-protocol voor communicatie tussen de client en de server. Bij iedere interactie zal de hele pagina verstuurd worden en moet de browser deze opnieuw renderen. Hierdoor krijgen we de bekende flikkering van het scherm. Voor statische webpagina's volstaat dit model. Echter, vandaag de dag worden we omringd door pagina's die heel wat interactie vragen van de gebruiker. Het wordt hoe langer hoe meer duidelijk dat er iets moet veranderen. Welkom in de wereld van Ajax.
AJAX, of voluit Asynchronous Javascript and XML is een verzameling van technologieën die de interactiviteit van webapplicaties moet verhogen. Op het Windows-platform wordt hiervoor intern gebruikgemaakt van het XMLHttpRequest-object. Dit object, aanwezig in elke moderne browser, maakt het mogelijk om 'out-of-the-band' http-requests te doen naar de server. Zo'n request wordt niet gelanceerd door de browser, maar door een script in de pagina. Het XMLHttpRequest-object treedt op als proxy: het lanceert de request naar de server, wacht op antwoord en verwerkt dit antwoord later via javascript in de pagina. Zo wordt het mogelijk om delen van de pagina afzonderlijk te vernieuwen.

Wat is ASP.NET AJAX?

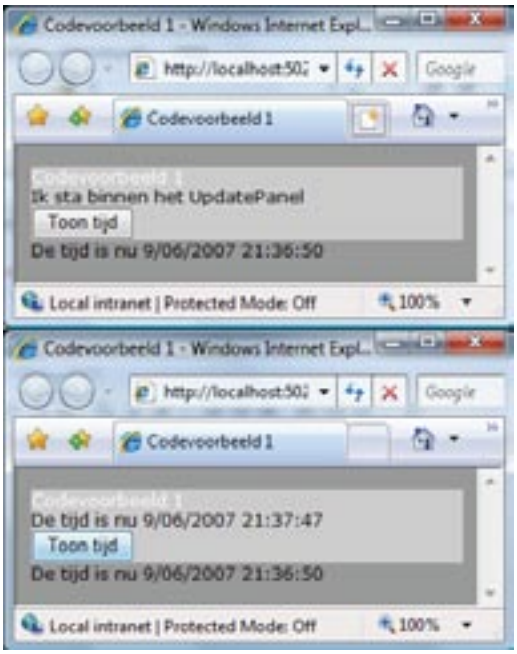
ASP.NET AJAX, vroeger bekend als Atlas, is Microsofts implementatie van de Ajax-technologie. Het is een framework dat zich zowel op de client als op de server uitstrekt, en is dus meer dan enkel een client-script library. Aan de client-side bestaat ASP.NET AJAX uit een client-script library: een grote verzameling van javascript-bestanden die kunnen worden gedownload naar de client. Deze library staat onder andere in voor grafische elementen en dient ook als trigger voor de partial page postbacks. Daarnaast levert ze aan Javascript heel wat uitbreidingen, zoals namespaces en een type-system. Hierdoor zal de scriptcode meer op .NET-code gaan lijken. Ten slotte regelt ze ook de cross-browser compatibiliteit.
Aan de server-side bevat het framework een aantal server-controls. Deze controls genereren naast standaard html ook nog scriptcode die naar de client meegestuurd wordt en voor de Ajax-functionaliteit zorgt. Andere nieuwigheden zijn de zogenaamde extenders, voornamelijk terug te vinden in de ASP.NET AJAX Control Toolkit. Deze server-controls geven aan bestaande ASP.NET-controls nieuwe client-side functionaliteit.

ASP.NET AJAX-extensions

Er zijn twee manieren om een applicatie met ASP.NET AJAX te bouwen: of we schrijven alle client-script code zelf, wat niet altijd even voor de hand liggend is voor de meeste ontwikkelaars, of we gebruiken speciale server-controls, die zelf scriptcode genereren. Zo kunnen we hetzelfde programmeermodel van ASP.NET 2.0 aanhouden. Bij deze laatste manier van werken, is er steeds een component aanwezig, die heel wat taken op zich neemt, zoals het sturen van de juiste scripts naar de client of partial page updates. In het geval van ASP.NET AJAX, is dit de ScriptManager. Elke pagina met Ajax-functionaliteit moet er precies één hebben. Deze server-control heeft geen visuele interface.
Het UpdatePanel maakt samen met de ScriptManager en de client PageRequestManager-class partial updates mogelijk. Hierdoor kunnen controls binnen een UpdatePanel asynchroon gaan posten naar de server. Terwijl zonder het UpdatePanel de hele pagina zou vernieuwen, stuurt nu het UpdatePanel een out-of-band request naar de server. Op de server wordt de volledige page-lifecycle doorlopen, maar alleen de html voor de UpdatePanels die aangepast moeten worden, wordt door de server teruggestuurd. Aan de client-side staat de PageRequestManager class in voor het vervangen van de oude markup in het Document Object Model (DOM) door de nieuwe.
Let wel, hoewel het UpdatePanel een container is voor andere controls, heeft deze geen UI-mogelijkheden zoals de standaard Panel. Een aantal van de belangrijkste eigenschappen:

- ChildrenAsTriggers: wanneer deze true is (standaard), wordt het UpdatePanel geüpdated zodra één van zijn childcontrols een postback triggert.
- UpdateMode: deze property bepaalt wanneer het UpdatePanel zal updaten. Als de waarde 'Always' is (standaard), wordt de content geüpdated bij elke postback. Dit gebeurt niet alleen bij postbacks van de pagina, maar ook bij postbacks die door andere UpdatePanels worden getriggerd. Bij de andere mogelijke waarde, 'Conditional', zal het UpdatePanel enkel aanpassen bij een postback die wordt veroorzaakt door een control die als trigger gedeclareerd is.

In codevoorbeeld 1 hebben we een UpdatePanel, met daarin een Button en een Label. Een tweede label bevindt zich buiten de panel. Wanneer de pagina voor het eerst laadt, wordt het Page_Load-event uitgevoerd en beide labels tonen een tekst. Bij het klikken op de Button, komt de partial page-update tot uiting. Dit is te zien aan het feit dat de browser geen progressiebalk toont. Hoewel de gehele page-lifecycle terug wordt uitgevoerd op de server, inclusief de Page_Load, wordt alleen de content van het UpdatePanel aangepast, zoals te zien op afbeelding 1.
Zonder aanpassing zal elk event van een control die zich in een UpdatePanel bevindt een Postback veroorzaken. Door allerhande redenen, meestal lay-out, kunnen we vaak de control die het UpdatePanel moet triggeren, niet plaatsen binnen de panel zelf. Daarom bestaan er triggers. In codevoorbeeld 2 staat de button nu buiten de UpdatePanel.



Afbeelding 1. UpdatePanel aan het werk: enkel de tijd binnen het UpdatePanel zal verversen.

Aan het UpdatePanel is nu een Triggers-tag toegevoegd. Een trigger specificeert dat een bepaald event van een control, in dit geval het Click-event van de button-control, het UpdatePanel zal verversen. Wanneer er meer UpdatePanels op een pagina staan, gaan deze allemaal vernieuwen bij elke postback, zelfs als deze veroorzaakt wordt door een control buiten de UpdatePanel, omdat de UpdateMode standaard de waarde Always heeft. Daarom kunnen we deze eigenschap op Conditional zetten: hierdoor zal het UpdatePanel alleen vernieuwen door het aanroepen van de Update-functie tijdens een postback of door één van zijn triggers. Om de gebruiker tijdens een postback feedback te geven over het feit dat een bepaalde functie veel tijd in beslag neemt, bestaat de UpdateProgress-control. Deze toont een boodschap zolang het UpdatePanel op een response van de server wacht. In codevoorbeeld 3 wordt deze control getoond.

Tot nu kon het UpdatePanel enkel maar verversen door interactie van de gebruiker. Door een Timer-control toe te voegen aan de pagina en deze te declareren als trigger voor de UpdatePanel, kunnen we de content op geregelde tijdstippen vernieuwen. De Timer-control definieert een Interval-property en een OnTick-event: na het verstrijken van het interval, zal het OnTick-event afgaan. In codevoorbeeld 4 wordt in dit event de source-naam van de te tonen afbeelding vervangen. Hierdoor krijgen we continu afwisselende afbeeldingen.

De ASP.NET AJAX Control Toolkit

UpdatePanels zijn vooral handig om partial page-updates te bereiken. Daarnaast kunnen we ook gebruikmaken van zogenaamde control extenders. Dit zijn server-controls die nieuwe client-side functionaliteit – ofwel behavior – geven aan een ASP.NET-control, door het toevoegen van script aan de gegenereerde opmaakcode. De ASP.NET AJAX Control Toolkit (ACT) is een verzameling van Ajax-enabled controls en extenders. Voor heel wat vaak voorkomende taken, is er al een extender meegeleverd (bijvoorbeeld AutoComplete, Drag and Drop, popups). De ACT wordt voornamelijk gebouwd door de community en is dus open-source. Recent werd het project verhuisd naar Codeplex en er worden geregeld nieuwe versies van uitgebracht. Een aantal voorbeelden van extenders:

- Autocomplete: deze extender maakt het mogelijk aan een TextBox een dropdown met suggesties toe te voegen
- DropShadowExtender: voegt schaduw toe aan nagenoeg elke control
- MaskedEdit: maakt enkel bepaalde input mogelijk in een TextBox

We gaan de werking van een extender verduidelijken aan de hand van een voorbeeld, namelijk de CollapsiblePanel. We starten hiervoor vanaf een 'Ajax Control Toolkit Website'. Deze template refereert automatisch al de assembly van de ACT en wijzigt de web.config. Om de controls en extenders van de toolkit toe te voegen aan de Toolbox in Visual Studio, moeten we simpelweg de 'AjaxControlToolkit.dll' selecteren in de 'Choose Toolbox Items'-dialog. Aangezien de ACT eigenlijk een library is, moeten we deze expliciet registreren op de pagina of in de web.config.
De CollaspsiblePanel is een extender waarmee je een dichtklapbaar panel kunt toevoegen aan webpagina's. Dit is bijzonder handig voor pagina's die veel informatie bevatten, de gebruiker kan zelf kiezen welke informatie getoond wordt op de pagina. Voor de werking ervan zijn twee panels nodig: een eerste panel bevat de eigenlijke content en een tweede dient als schakelaar voor het open- en dichtklappen. Om een CollapsiblePanelExtender te laten werken, moeten we nog een aantal properties opgeven, zoals in codevoorbeeld 5 is te zien.
De TargetControlId specificeert wat het dichtklapbaar panel wordt. De ExpandControlId en de CollapseControlId geven aan welke

```
protected void Page_Load(object sender, EventArgs e){
   Label1.Text = "Ik sta binnen het UpdatePanel";
   Label2.Text = "De tijd is nu " + DateTime.Now.ToString();
}
protected void Button1_Click(object sender, EventArgs e){
   Label1.Text = "De tijd is nu " + DateTime.Now.ToString();
}
...
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
   <ContentTemplate>
      <asp:Label ID="Label1" runat="server"></asp:Label><br />
      <asp:Button ID="Button1" runat="server" Text="Toon tijd"
           OnClick="Button1_Click" />
   </ContentTemplate>
</asp:UpdatePanel>
<asp:Label ID="Label2" runat="server"></asp:Label>
```

Codevoorbeeld 1.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
   <Triggers>
      <asp:AsyncPostBackTrigger ControlID="Button1" EventName="Click" />
   </Triggers>
</asp:UpdatePanel>
```

Codevoorbeeld 2.

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server"
        AssociatedUpdatePanelID="UpdatePanel1">
   <ProgressTemplate>
   <img src="loading.gif" /> Bezig met laden...
   </ProgressTemplate>
</asp:UpdateProgress>
```

Codevoorbeeld 3.

```
<asp:Timer ID="Timer1" runat="server" Interval="3000"
     OnTick="Timer1_Tick" >
</asp:Timer>
   <asp:UpdatePanel ID="UpdatePanel1" runat="server">
      <ContentTemplate>
         <asp:Image ID="Image1" runat="server" ImageUrl="banner1.gif" />
      </ContentTemplate>
   <Triggers>
      <asp:AsyncPostBackTrigger ControlID="Timer1" />
   </Triggers>
</asp:UpdatePanel>
```

Codevoorbeeld 4.

controls dit moeten triggeren. Ook kunnen we specificeren dat een bepaald tekstlabel, in dit geval Label1 moet wijzigen naargelang de toestand van het panel. Het open- en dichtklappen gebeurt client-side, dus zonder postback naar de server. Bij een eventuele round-trip naar de server door een actie op een andere control, bewaart het CollapsiblePanel wel zijn state.

Zelf extenders maken

De class ExtenderControl, gedefinieerd door de ASP.NET AJAX-extensions, is de base class voor alle extenders, ook diegene in de ACT. Deze class erft over van Control en implementeert IExtenderControl. Zoals al eerder is beschreven, voegt elke extender een bepaalde functionaliteit toe aan een bepaalde control. De koppeling tussen extender en control gebeurt door de TargetControlID-property, gedefinieerd in ExtenderControl.
In het volgende voorbeeld gaan we een vereenvoudigde versie van de TextBoxWatermark-extender, aanwezig in de ACT, schrijven. De TextBox toont een standaard tekst, totdat de gebruiker in het veld klikt. Dit kan gebruikt worden als vervanging van tooltips. Om deze extender te bouwen, vertrekken we van de 'ASP.NET AJAX Control Project'-template. We noemen de control 'MSDNTextBox'. Er worden drie bestanden gegenereerd:

- MSDNTextBoxBehavior.js: deze file bevat het script dat uitgevoerd wordt op de client.
- MSDNTextBoxDesigner.cs: in deze file kunnen we design-time functionaliteit toevoegen.
- MSDNTextBoxExtender.cs: dit is de eigenlijke server-side control class. Hierin staan ook de eigenschappen van de extender gedeclareerd, die overeenkomen met eigenschappen in de client-script file.

```
<ajaxToolkit:CollapsiblePanelExtender ID="CollapsiblePanelExtender1"
   runat="server" TargetControlID="pnlInhoud"
   ExpandControlID="pnlHoofding" CollapseControlID="pnlHoofding"
   Collapsed="true" TextLabelID="Label1" ImageControlID="Image1"
   ExpandedText="(Toon grid...)" CollapsedText="(Verberg grid...)"
   SuppressPostBack="true">
</ajaxToolkit:CollapsiblePanelExtender>
```

Codevoorbeeld 5.

```
[TargetControlType(typeof(TextBox))]
public class MSDNTextBoxExtender : ExtenderControlBase{
   [ExtenderControlProperty]
   public string DefaultText
   {
        get { return GetPropertyValue("DefaultText", ""); }
        set { SetPropertyValue("DefaultText", value); }
   }
}
```

Codevoorbeeld 6.

```
MSDNTextBox.MSDNTextBoxBehavior = function(element) {
   MSDNTextBox.MSDNTextBoxBehavior.initializeBase(this, [element]);
   this._DefaultTextValue = null;
   this._clickHandler = null;}
```

Codevoorbeeld 7.

```
get_DefaultText : function() { return this._DefaultTextValue;},
set_DefaultText : function(value) { this._DefaultTextValue = value;}
```

Codevoorbeeld 8.

```
initialize : function() {
   MSDNTextBox.MSDNTextBoxBehavior.callBaseMethod(this, 'initialize');
   this.get_element().value = this._DefaultTextValue;
   this._clickHandler = Function.createDelegate(this, this._onClick);
   $addHandler(this.get_element(), "click", this._clickHandler);
},
_onClick : function(e) {
   if(this.get_element().value == this._DefaultTextValue){
     this.get_element().value = "";
   }
}
```

Codevoorbeeld 9.

```
<%@ Register Assembly="MSDNTextBox" Namespace="MSDNTextBox"
TagPrefix="cc1"%>
...
<asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
<cc1:MSDNTextBoxExtender ID="MsdnTB1" runat="server"
   TargetControlID="TextBox1" DefaultText="enter text">
</cc1:MSDNTextBoxExtender>
```

Codevoorbeeld 10.

We beginnen onze aanpassingen in de eigenlijke Extender-file, MSDNTextBoxExtender.cs. Op de class is een aantal attributen gedeclareerd. Eén daarvan is deTargetControlType die standaard de waarde Control heeft. Dit betekent dat deze extender op elk type van control kan worden toegepast. In ons geval is de extender alleen van toepassing op een TextBox. In codevoorbeeld 6 is dit attribuut aangepast. In de Extender maken we nu ook een nieuwe eigenschap, nl DefaultText, de initiële tekst in de TextBox. Deze eigenschap moet ook verder in de scriptfile aangemaakt worden. Het grootste werk situeert zich in de scriptfile: we zullen zelf client-functionaliteit toevoegen. In de 'initializeBase'-functie voegen we twee nieuwe variabelen toe: _DefaultTextValue om de waarde die via de extender doorgegeven wordt, te bewaren, en _clickHandler, een delegate. Verder hebben we accessor-functies nodig, terug te vinden in codevoorbeeld 8.
In de functie initialize gaan we eerst de waarde die aan de extender wordt doorgegeven via de DefaultText-eigenschap, weergeven in de TextBox waaraan de extender gekoppeld is. De belangrijkste functie is de client-side-code om een click op te vangen in een TextBox. De declaratie ervan plaatsen we mee in de functie initialize. In de uitwerking van de functie controleren we of de huidige waarde van het veld gelijk is aan de opgegeven waarde. Als dit het geval is, wordt het veld leeggemaakt en kan de gebruiker zelf een waarde ingeven. Zie hiervoor codevoorbeeld 9. De implementatie van de extender is hiermee afgerond. We kunnen deze nu gaan testen in een webapplicatie. Om de extender te testen, moeten we eerst een Register-page-directive toevoegen op de pagina. Hierdoor kan onze control binnen deze pagina worden geïnstantieerd. Verder hebben we een TextBox nodig, waarop onze extender zal werken, en natuurlijk een instantie van de extender zelf, zoals te zien op codevoorbeeld 10. De koppeling gebeurt door de TargetControlId.

Toekomst

ASP.NET AJAX is de toekomst van ASP.NET, en is daarmee zeker een interessant platform om te volgen. In dit artikel heb ik de belangrijkste facetten van ASP.NET AJAX kort belicht en er is dan ook nog heel wat zelf te ontdekken; denk maar aan nieuwe releases van de toolkit. Het is dan ook heel interessant om te zien hoe Ajax zal integreren in volgende versies van de core van ASP.NET.

Gill Cleeren is senior developer bij Ordina Belgium (www.ordina.be) en leidt het Task Force Custom Development. Als early adopter is hij steeds bezig met de technologie van morgen. Meer is te vinden op zijn blog www.snowball.be .

Referenties

ASP.NET AJAX: http://ajax.asp.net
AJAX Control Toolkit: http://www.codeplex.com/AtlasControlToolkit
Scott Guthrie over AJAX: http://weblogs.asp.net/scottgu/archive/tags/Atlas/default.aspx