Dennis van de Laar

ASP.NET AJAX-control bouwen

VOLLEDIGE VRIJHEID EN CONTROLE OVER DE ASYNCHRONE COMMUNICATIE VAN EEN WEBPAGINA

De meeste .NET-developers zijn bekend met het bouwen van eigen controls - webcontrols, usercontrols. Het is ook mogelijk eigen ASP.NET AJAX-controls te bouwen door gebruik te maken van de scriptlibrary MicrosoftAjax.js. Het bouwen van eigen controls is niet alleen cool, maar ook erg nuttig. Naast de vrijheid die je hebt, valt er veel voordeel te behalen door efficiënter om te gaan met postbacks.

Iedere ontwikkelaar die met ASP.NET AJAX heeft gewerkt, is bekend met de serverside-control genaamd 'UpdatePanel'. Een updatepanel is een container-control die ervoor zorgt dat de controls die hij bevat asynchroon communiceren met de server. Hierdoor zal de gebruiker ervaren dat de website blijft reageren, wanneer hij een actie heeft uitgevoerd. De data die worden verstuurd bij de asynchrone communicatie zijn echter niet zo efficiënt. Bij een asynchrone postback worden dezelfde data verstuurd als bij een standaard postback. De levensloop van de pagina is vrijwel dezelfde, alleen de rendering van de pagina is anders. De control die binnen het updatepanel valt, zal enkel worden geüpdatet.
Je zult bij elke AJAX-gerelateerde oplossing moeten nadenken over wat je wilt bereiken en een overweging maken of de standaard serverside-controls daarvoor geschikt zijn. Als dit niet het geval is, kun je de scriptlibraries van het ASP.NET AJAX-framework gebruiken om zelf ASP.NET AJAX-controls te bouwen. Een situatie waarin het bouwen van een eigen control overwogen kan worden, is wanneer een webapplicatie communiceert met een webservice. Op het internet zijn genoeg artikelen te vinden die beschrijven hoe een AJAX-control gebouwd moet worden. In dit artikel wil ik voornamelijk laten zien wat er binnen het framework wordt gedaan om een control te laten functioneren. Ik geef een korte toelichting over hoe je een dergelijke control maakt en de keuzes die gemaakt moeten worden, maar de focus ligt op hoe het ASP.NET AJAX-framework omgaat met jouw control. Als je een ASP.NET AJAX-control wilt maken, moet je bekend zijn met objectgeoriënteerde (OO) ontwikkeling binnen het ASP.NET AJAX-framework (Javascript). Hierbij moet je denken aan het registreren van namespaces, classes en interfaces. Als je een AJAX-control bouwt, kun je dit doen in de webpagina waar je hem gaat gebruiken. Het is echter verstandiger om de control te ontwikkelen in een apart Javascript-bestand voor beter hergebruik en onderhoud.

Ajax-control

In codevoorbeeld 1 zie je een voorbeeld van een simpele control waarin een button is toegevoegd die een click-event heeft. Bovendien is er een mouseover-event toegevoegd aan de control. Later in het artikel wordt duidelijk hoe het ASP.NET AJAX-framework omgaat met deze control. In codevoorbeeld 1 is ook te zien dat er gebruik wordt gemaakt van een prototype-object. Een prototype kun je zien als een baseclass van welke alle instanties overerven. In een prototype dien je de properties en functies van de control te definiëren.

Initialize-functie

De initialize-functie binnen een control is van groot belang als je events wilt gebruiken. Dit geldt zowel voor events die op de control plaatsvinden als voor events die op DOM-elementen binnen de control plaatsvinden. Bovendien kun je in de initialize-functie nieuwe DOM-elementen creëren en CSS-styles toepassen op de control. In codevoorbeeld 2 is een button aan de control toegevoegd en een eventhandler aan het onclick-event gekoppeld. Ook wordt een delegate gekoppeld aan het mouseover-event.

Dispose-functie

In de dispose-functie moet je alle delegates en handlers verwijderen die zijn toegevoegd in de initialize-functie. In codevoorbeeld 3 staat een voorbeeld voor het verwijderen van de handler die is gekoppeld aan het mouseover-event. Dit voorbeeld voeg je toe aan de dispose-functie binnen de control. Er is binnen het ASP.NET AJAX-framework ook een verkort commando $clearhandlers. Deze functie zorgt er voor dat alle gedefinieerde handlers worden verwijderd. Hierdoor hoef je niet bij te houden welke handlers je hebt gekoppeld aan events. Deze worden gebruikt in codevoorbeeld 1.
Nadat we in vogelvlucht een control hebben gemaakt, gaan we nu kijken hoe het ASP.NET AJAX-framework hier mee omgaat.

ASP.NET AJAX-framework en de control

Wat gebeurt er nu precies binnen het framework om de gebouwde control op de juiste wijze te initialiseren? Hoe wordt omgegaan met properties, events, functies en referenties? Allereerst moet je een referentie opgeven naar het Javascript-bestand waarin de control is gebouwd. De referentie moet je opgeven bij de scriptmanager. De scriptmanager is een control die altijd op een webpagina aanwezig moet zijn als je gebruikmaakt van ASP.NET AJAX-functionaliteit. Een van de verantwoordelijkheden van deze control is het laden van de juiste scriptlibraries en dus ook de scriptlibrary van de controls. Een referentie kun je op meer manieren opgeven. In codevoorbeeld 4 is een mogelijkheid weergeven.

Nu de scriptmanager van het bestaan van de control weet kan de control gecreëerd worden. Dit wordt gedaan via de $create-functie. Dit is een verkort commando van de functie 'Sys$Component$create(type, properties, events, references, element)'. In codevoorbeeld 5 wordt gebruikgemaakt van de $create-functie. Dit gebeurt wanneer de applicatie wordt geladen.

$create(type, properties, events, references, element)

De $create-functie is de functie binnen het ASP.NET AJAX-framework die er voor zorgt dat de control geïnstantieerd wordt. Binnen deze functie worden andere functies aangeroepen die verantwoor-

```
// Registreer een namespace waarin de control beschikbaar wordt gesteld
Type.registerNamespace("Avanade");

// Constructor, welke een element (DOM-element) als input parameter
// verwacht. Het element zal worden gebruikt als host voor je control.
Avanade.Control = function(element) {
   Avanade.Control.initializeBase(this, [element]);
   this._mouseoverDelegate = null;
   this._name = null;
}

// Een prototype waarin properties en functies worden gedefinieerd.
Avanade.Control.prototype = {
   // Een propertie welke get_ en set_ als prefix heeft, dit is
   // noodzakelijk anders herkend het ASP.NET AJAX framework de propertie niet
   get_name: function() {
      return this._name;
   },
   set_name: function(value) {
      this._name = value;
   },

   // Het toevoegen en verwijderen van een handler aan de Dictionary van
   // je control. Voor het toevoegen van eventhandlers wordt de add_ en
   // remove_ prefix gebruikt
   add_mouseover: function(handler) {
      this.get_events().addHandler('mouseover', handler);
   },
   remove_mouseover: function(handler) {
      this.get_events().removeHandler('mouseover', handler);
   },

   // Een functie die in de initialize functie gekoppeld wordt
   // aan een nieuwe button
   _handleclick: function()
   {
   alert('handleclick');
   }
   ,

   // In de initialize functie kan je DOM elementen toevoegen aan je
   // control en events koppelen aan DOM elementen en aan de control zelf
   initialize: function() {

      var element = this.get_element();

      var button = document.createElement('button');
      button.appendChild(document.createTextNode('button1'));

      button.onclick = this._handleclick;
      element.appendChild(button);

      if (!element.tabIndex) element.tabIndex = 0;

      if (this._mouseoverDelegate === null) {
         this._mouseoverDelegate = Function.createDelegate(this,
         this._mouseoverHandler);
      }
      $addHandler(element, 'mouseover', this._mouseoverDelegate);

      Avanade.Control.callBaseMethod(this, 'initialize');
   },

   // In de dispose functie moet je alle eventhandlers en delegates
   // verwijderen welke je hebt aangemaakt in de initialize functie.
   dispose: function() {

      var element = this.get_element();

      if (this._mouseDelegate) {
         delete this._mouseDelegate;
      }

      // $clearhandlers wordt gebruikt om alle eventhandlers die
      // gekoppeld zijn aan de control te verwijdereren. Je dient de
      // control als input parameter mee te geven.
      $clearHandlers(element);

      Avanade.Control.callBaseMethod(this, 'dispose');
   },

   // Functie die eventhandlers uit de lijst van events haalt
   // a.d.h.v. een key(eventnaam)
   _mouseoverHandler: function(event) {
      var h = this.get_events().getHandler('mouseover');
      if (h) h(this, Sys.EventArgs.Empty);
   }
}

// De control wordt geregistreerd in de namespace en erft over van
// Sys.UI.Control
Avanade.Control.registerClass('Avanade.Control', Sys.UI.Control);

// Met dit statement geef je aan de scriptmanager aan dat het script
// vanje control is gedownload.

if (typeof(Sys) !== 'undefined') Sys.Application.notifyScriptLoaded();
```

Codevoorbeeld 1. Een voorbeeld van een simpele control

```
initialize: function() {

   // Element vertegenwoordigt het control
      var element = this.get_element();

   // Maak een nieuwe button
      var button = document.createElement('button');
   button.appendChild(document.createTextNode('button1'));

   // Voeg handleclick functie toe als eventhandler van button click
   button.onclick = this._handleclick;

   // Voeg button toe aan het control
   element.appendChild(button);

      if (!element.tabIndex) element.tabIndex = 0;

   // Maak een delegate die naar de eventhandler mouseoverHandler refereert
      if (this._mouseoverDelegate === null) {
         this._mouseoverDelegate = Function.createDelegate(this,
         this._mouseoverHandler);
      }

   // Koppel de delegate aan het mouseover event
   $addHandler(element, 'mouseover', this._mouseoverDelegate);

      Avanade.Control.callBaseMethod(this, 'initialize');
   },
```

Codevoorbeeld 2. Toevoegen van een DOM element en events

delijk zijn voor het toevoegen van properties, events en referenties. De properties, events en referenties worden door middel van JavaScript Object Notation (JSON) gestructureerd. JSON is een tekstformaat waarin data worden gestructureerd om deze te gebruiken voor de communicatie tussen client en server. In afbeelding 1 zie je de structuur die ook wordt gebruikt in het create-statement in codevoorbeeld 5. De parameters 2 tot en met 4 worden volgens deze structuur opgebouwd.

Door middel van JSON kun je eenvoudig een lijst met properties, functies en referenties op gestructureerde wijze aan je control aanbieden.

In codevoorbeeld 5 is een DIV-element ($get('ControlDiv')) meegegeven in de create-functie. De $get-functie is een verkort commando voor document.getElementById(). Houd er dus rekening mee dat er wel een div op de webpagina moet staan met in mijn geval een id ControlDiv. Mijn DIV-element zal als input-parameter worden gebruikt in de constructor van de control. Het element zal dus de control gaan vertegenwoordigen op de webpagina. Nadat er controles zijn uitgevoerd op de meegegeven parameters wordt er in de create-functie een aanroep gedaan naar de constructor van de control. In codevoorbeeld 6 is te zien dat een if-else-statement wordt uitgevoerd. Als het element bestaat, dan wordt het element als input-parameter gebruikt voor de constructor van de control. De eerste parameter van het $create-statement is 'type' oftewel de control die je wilt initialiseren. Uiteindelijk resulteert dit in een component die verderop in het create-proces verrijkt kan worden met properties, events en referenties.

```
$removeHandler(element, 'mouseover', this._mouseoverDelegate);
```

Codevoorbeeld 3.

```
<Scripts>
     <asp:ScriptReference Path="Javascript/Control.js" />
</Scripts>
```

Codevoorbeeld 4. Het opgeven van een referentie

```
<script type="text/javascript">

  // Koppel de applicationLoadHandler aan de load van de
  // webapplicatie
  Sys.Application.add_load(applicationLoadHandler);

  function applicationLoadHandler(sender, args)
  {
  // create functie met 5 parameters (JavaScript Object Notation)
  $create(Avanade.Control,
  {name: 'newControl'},
  {mouseover: DoMouseover},
  null,
  $get('ControlDiv'));
  }

  function DoMouseover(sender, args) {
  alert("Mouse over the control");
  }
</script>
```

Codevoorbeeld 5. Het gebruik van de $create-functie

Sys$Component$_setProperties (target, properties);

Nu de control binnen het ASP.NET AJAX-framework bekend is, kunnen de properties worden gebruikt om de control van informatie te voorzien. De parameters die de setProperties-functie ontvangt, zijn de control zelf (target) en de properties die zijn meegegeven in het $create-statement. In mijn geval is dat enkel een naam (name). De properties worden als een lijst van key-value-pairs doorlopen (JSON). Voor iedere property wordt een functie gezocht binnen de control. In codevoorbeeld 7 wordt een setter-functie gezocht binnen de control: "set_" + de naam van de desbetreffende property (set_name). Vervolgens wordt gecontroleerd of de setter van de control gedefinieerd is als een functie. Als dit niet zo is, dan wordt de property niet gebruikt. Is dit wel zo, dan zal deze setter worden uitgevoerd door een target (de control) en een value ('newControl') meegeven. De scriptlibrary weet hierdoor welke property hij van welke control moet benaderen. Als je hier met de debugger in stapt, dan kom je bij de property in codevoorbeeld 8. Zo worden alle properties gezet die worden opgegeven in het $create()-statement.

Events

Nadat de properties voor de control zijn verwerkt, zal het framework de events verwerken. Events in een ASP.NET AJAX-control zijn de meest complexe binnen de control. Als je met events gaat werken, is het belangrijk dat je begrijpt binnen welke context events worden uitgevoerd. Het woord 'this' kan heel erg verwarrend werken. De ene keer vertegenwoordigt 'this' het element dat het event triggered, de andere keer is het de control of het window-object. Dit kan leiden tot onverklaarbaar gedrag wat lastig op te lossen is als je niet weet binnen welke context bepaalde acties worden uitgevoerd.

In codevoorbeeld 9 is 'this' de control en element is het html-element. Als je optie 1 uitvoert, wordt de handler toegevoegd aan het element en als je optie 2 uitvoert, wordt de handler toegevoegd aan de control, waardoor 'this' naar verschillende objecten wijst. Bij optie 2 wordt gebruikgemaakt van de functie createDelegate. Als

```
// De notitie in het ASP.NET AJAX framework
var component = (element ? new type(element): new type());

// In mijn situatie zal het als volgt geïnterpreteerd worden
var component = (element ? new Avanade.Control(element): new Avanade.
Control());
```

Codevoorbeeld 6. Het uitvoeren van een if-else-statement

```
// prefix set_ is noodzaklijk wanneer je properties definiëerd, omdat
het framework
// daar vanuit gaat.
var setter = target["set_" + name];
     if (typeof(setter) === 'function') {
         setter.apply(target, [val]);
}
```

Codevoorbeeld 7. Een setter-functie wordt gezocht binnen de control

```
set_name: function(value) {
     this._name = value;
   }
```

Codevoorbeeld 8.

je van deze functie gebruikmaakt, ben je tijdens de initialisatie van de control in staat aan te geven binnen welke context een functie uitgevoerd moet worden. Bij optie 2 wordt 'this' meegegeven als de context waarbinnen het event uitgevoerd dient te worden. Dit houdt in dat het mouseover-event binnen de context van de control wordt uitgevoerd. In afbeelding 2 is te zien dat 'this' het html-element vertegenwoordigt. In afbeelding 3 is te zien dat 'this' de control vertegenwoordigt.
Binnen een control zijn er twee opties om eventhandlers aan events te koppelen:

- Eventhandlers zelf definiëren binnen de control en koppelen aan de events
- Eventhandlers door de gebruikers van de control laten definiëren en aan de events koppelen
-

Hier dien je goed over na te denken. Je moet jezelf afvragen of je de gebruiker van jouw custom AJAX-control de vrijheid wilt geven om zelf functies aan events te koppelen die plaatsvinden op jouw control. Als je die vrijheid wilt bieden, moet je twee extra stappen ondernemen:

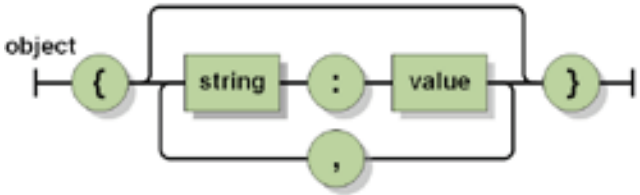
- Creëer twee functies om handlers toe te voegen en te verwijderen (Add_[handler] en Remove_[handler])
- Creëer een functie die handlers uitvoert wanneer het event plaatsvindt

De drie functies die gebouwd moeten worden, moeten gebruikmaken van een EventHandlerList die beschikbaar is binnen de control. De EventHandlerList bevat drie functies:

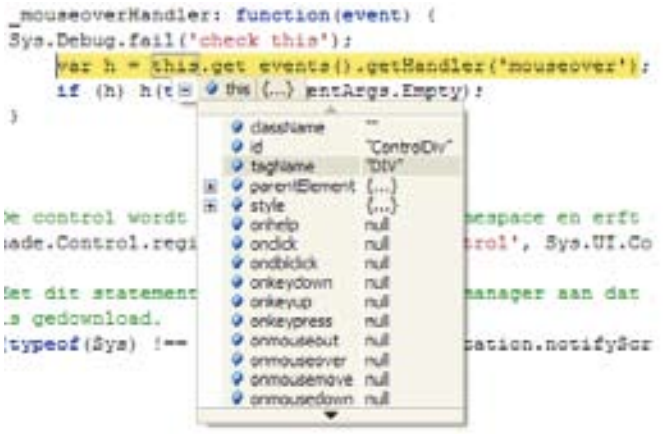
- addHandler (toevoegen van een handler en een event aan de list)
- removeHandler (verwijderen van een handler en een event uit de list)
- getHandler (geeft alle handlers die gekoppeld zijn aan een event).

Deze list wordt alleen gebruikt voor de registratie van events en eventhandlers.

In codevoorbeeld 10 is te zien hoe je twee functies maakt die gebruikers van de control in staat stellen events en eventhandlers toe te voegen aan de Eventhandlerlist. In codevoorbeeld 11 is te zien hoe eventhandlers uit de list worden gehaald aan de hand van de naam van een event. Binnen de list kunnen meer eventhandlers aan een event zijn gekoppeld. Door op deze manier eventhandlers op te zoeken en uit te voeren, kun je op een simpele manier verscheidene eventhandlers uitvoeren wanneer een event wordt getriggerd. Nu eventhandlers toegevoegd kunnen worden door gebruikers van de control en de control in staat is aan de hand van de eventnaam de juiste eventhandler(s) uit de EventHandlerList te halen en uit te voeren, moet je enkel nog de eventhandler aan het event van de control koppelen. Dit is een actie die ook



Afbeelding 1. Schematische weergave van de structuur van JSON



Afbeelding 2. 'this' vertegenwoordigt het html-element

uitgevoerd moet worden wanneer je zelf eventhandlers binnen jouw control aan events koppelt. Het koppelen van eventhandlers aan events van een DOM-element wordt per browser verschillend uitgevoerd. In tabel 1 zie je hoe browsers omgaan met events en eventhandlers.
De verschillen in interpretatie van Javascript door de browsers kan voor ontwikkelaars voor extra uitdagingen zorgen. Hoe koppel je een click-eventhandler aan een button die werkt in IE en Firefox? Als je gebruikmaakt van het ASP.NET AJAX-framework hoef je je daar niet druk om te maken. Dit is allemaal voor je geregeld. Binnen het ASP.NET AJAX-framework wordt 'multi-browser support' voor je geregeld door gebruik te maken van de $addhandler()-functie. $addhandler is een verkort commando (Sys.UI.DomEvent.addHandler) om events aan elementen toe te voegen. Binnen deze functie wordt gecontroleerd in welke browser de code wordt uitgevoerd. Afhankelijk van het resultaat wordt de syntax gebruikt die door de desbetreffende browser wordt ondersteund.De $addhandler-functie verwacht drie parameters:

- Het element dat het event afvuurt
- Het event dat zal plaatsvinden (geen onclick, maar 'click')
- Een delegate / functie die als eventhandler zal fungeren

Binnen de $addHandler-functie worden dus alle afhankelijkheden afgevangen die staan beschreven in tabel 1. Het is daarom verstandig altijd gebruik te maken van de $addHandler-functie wanneer je events koppelt aan eventhandlers. Het koppelen van events aan eventhandlers gebeurt altijd in de initialize-functie; zie codevoorbeeld 1. Als je ervoor hebt gekozen om de gebruikers van

```
// this.get_element zal verwijzen naar de control zelf aangezien deze
// code in de initialize functie van de control moet staan.
var control = this.get_element();

    // Hier wordt een handler gekoppeld aan de mouseover event
    // van het element.
  1. $addHandler(element, 'mouseover', this._mouseoverHandler);
    // Hier wordt een handler gekoppeld aan de mouseover event van de
    // control d.m.v. een delegate.
  2. this._mouseoverDelegate = Function.createDelegate(this, this._mo
useoverHandler);
    $addHandler(element, 'mouseover', this._mouseoverDelegate);
```

Codevoorbeeld 9.

```
// Het toevoegen en verwijderen van een handler aan de list van je control.
// Voor het toevoegen van eventhandlers wordt de add_ en
// remove_ prefix gebruikt
add_mouseover: function(handler) {
  this.get_events().addHandler('mouseover', handler);
},
remove_mouseover: function(handler) {
  this.get_events().removeHandler('mouseover', handler);
},
```

Codevoorbeeld 10. Het maken van twee functies

	IE	Firefox, Opera, Safari
Toevoegen van een event	attachEvent	AddEventListeners
Naam event	On + eventnaam (onclick)	Eventnaam (click)
Event parameter	Window.event (in IE 7 ook beschikbaar als parameter)	Als parameter meegegeven

Tabel 1. Browser-verschillen

de control de mogelijkheid te bieden eventhandlers toe te voegen aan events die plaatsvinden op de control, dan dienen zij de events en handlers te declareren in het $create-statement van de control. Ze geven hierbij de naam van het event en de naam van de eventhandler op die zij zelf hebben gemaakt (derde parameter van het $create-statement). In codevoorbeeld 12 is te zien hoe het ASP.NET AJAX-framework door de lijst van events heenloopt om de functies binnen de control te benaderen. Tijdens de initialisatie van de control worden de eventhandlers die zijn opgegeven door de gebruikers van de control aan de events gekoppeld.

Sys$Component$_setReferences (component, references);

Tot slot is het mogelijk referenties naar andere controls toe te voegen aan jouw control. Dit gebeurt op dezelfde wijze als het toevoegen van properties. Er moeten functies in de control gedefinieerd worden die het toevoegen en opvragen van referenties mogelijk maakt. Beide functies moeten van 'get_' en 'set_' prefixes worden voorzien. Als er tijdens het aanmaken van een control een referentie wordt opgeven naar een andere control, dan zal de for-loop in codevoorbeeld 13 worden doorlopen. Hierin is te zien dat wordt gezocht naar een functie binnen jouw control genaamd: set_ + [naam reference]. Vervolgens wordt de referentie naar een andere control gelegd. Hiervoor wordt gebruikgemaakt van de functie $find. Deze functie biedt je dezelfde functionaliteit als $get, alleen zoekt $find niet op het ID van DOM-elements (DIV, Buttons, enzovoort) maar op het ID van components (Controls). Als de component beschikbaar is, wordt de setter-functie uitgevoerd en wordt de gevonden component toegevoegd als referentie aan de control.

Samenvatting

Kortom, het schrijven van jouw eigen Ajax-control biedt heel veel mogelijkheden. Je hebt de volledige vrijheid om de control zo in te richten dat deze voldoet aan al jouw wensen. In dit artikel heb ik laten zien hoe het framework is ingericht en waar ontwikkelaars met ASP.NET AJAX rekening mee moeten houden wanneer een control wordt gebouwd. Ik heb beschreven hoe en waarom je properties, functies en referenties binnen een Ajax-control definieert. Ook heb ik laten zien hoe je omgaat met events en de context waarbinnen de eventhandlers worden uitgevoerd. Het bouwen van

Afbeelding 3. 'this' vertegenwoordigt de control

```
// Functie die eventhandlers uit de lijst van events haalt
// a.d.h.v. een key (event-naam)
_mouseoverHandler: function(event) {
    var h = this.get_events().getHandler('mouseover');
    if (h) h(this, Sys.EventArgs.Empty);
  }
```

Codevoorbeeld 11. Eventhandlers worden uit de list gehaald

```
// een foreach loop binnen de events lijst.
// Ik had maar 1 event opgegeven wat zal
// resulteren in: component[add_mouseover]
// (DoMouseover); hierdoor wordt add_mouseover
// functie aangeroepen en DoMouseover toegevoegd
// aan de EventHandlerlist.
for (var name in events) {
    component["add_" + name](events[name]);
  }
```

Codevoorbeeld 12. De lijst met events wordt doorlopen

```
for (var name in references) {
    var setter = component["set_" + name];
    var reference = $find(references[name]);
    setter.apply(component, [reference]);
  }
```

Codevoorbeeld 13. De for-loop wordt doorlopen

een eigen ASP.NET AJAX-control vergt kennis van Javascript. Overweeg een eigen control te maken, wanneer je volledige controle wilt over het gedrag van jouw webpagina en dan voornamelijk de asynchrone communicatie.

Dennis van de Laar is als Senior Associate Consultant werkzaam bij Avanade (www.avanade.com), een samenwerkingsverband tussen Microsoft en Accenture. Voor vragen of opmerkingen is hij te bereiken op dennisv@avanade.com of bekijk zijn blog http://dennisv.net

Referenties

Voor documentatie over het ASP.NET AJAX-framework: http://asp.net/ajax/documentation/live/

Voor informatie over JSON: www.json.org

Een goed voorbeeld hoe het updatepanel inefficiënt omgaat met viewstate: http://encosia.com/index.php/2007/07/11/why-aspnet-ajax-updatepanels-are-dangerous/