Silverlight 2 Communicator Web Access Client

Hoe maak je zo'n instant messaging client?

Marc Wetters

Sinds oktober 2008 is Silverlight 2 beschikbaar. Met deze nieuwe versie van Silverlight kun je een instant messaging client maken, die communiceert met de Communicator Web Access-server. Hoe gaat dat precies?

In een ideale situatie draait een Silverlight-applicatie in elke webpagina op een willekeurige server. De grootste uitdaging is cross-domain-toegang. Binnen bepaalde randvoorwaarden ondersteunt Silverlight dit, hoewel alleen met http-connecties en nog niet met https. Daarnaast moet op de target server een XML file beschikbaar zijn die Silverlight toestaat de server te benaderen. Omdat veiligheid bij businesscommunicatie voorop staat, moet de applicatie gehost worden op de CWA-server.

Voor het ontwikkelen van de applicatie is https geen vereiste. Dus het plaatsen van de XML file op de server zou moeten voldoen. Alleen de CWA isapi-filter vangt van de root alles af. Daarnaast bestaat er nog een reden om de applicatie altijd op de CWA-server te draaien. Hierop gaan we verder in onder het kopje 'authenticatie'.

Voor een ontwikkelaar levert dit meteen een extra uitdaging op. Wat voor ontwikkelomgeving heb je nodig om ook debuggen mogelijk te maken? Dit gaat alleen als de ontwikkelomgeving direct op een CWA-server draait. En wel een CWA-server met Visual Studio 2008. Belangrijk hierbij is dat we de webapplicatie binnen de CWA-omgeving opzetten en deze ook gebruiken bij de start van het Visual Studio 2008-project (zie afbeeldingen 1 en 2)

Het probleem van debuggen is opgelost. Nu komt de grootste uitdaging.

Authenticatie

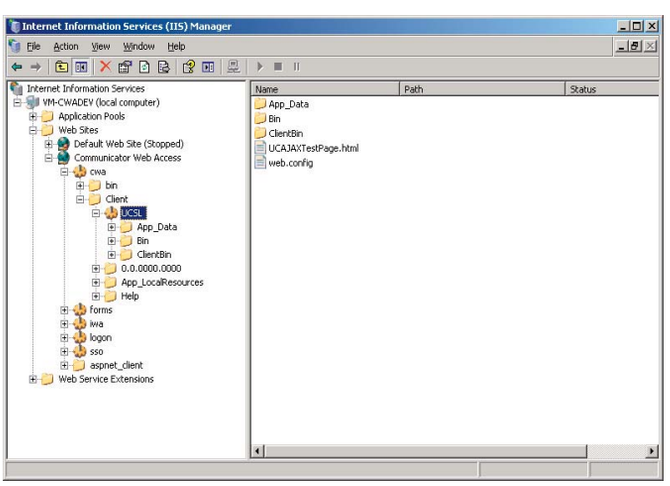
De CWA Ajax SDK kent verschillende manieren om te verifiëren (forms-authenticatie en Integrated authenticatie). Ze gebruiken een http-header in de response met daarin een authenticatie-ticket (CWA-ticket). Na succesvol verifiëren (via forms of via integrated authenticatie) genereert de CWA-server dit CWA-ticket. Het CWA-ticket is nodig voor alle verdere communicatie met de CWA-server.

Helaas kent het System.Net.HttpWebResponse-object van Silverlight geen http-headers. De eerste gedachte dat het niet gaat lukken, blijkt onterecht te zijn. We kunnen namelijk het XMLHttpRequest-object van de browser gebruiken. Deze ondersteunt wel http-headers - de tweede reden om op de CWA-server te ontwikkelen. Het XMLHttpObject ondersteunt geen cross domain access (zie codevoorbeeld 1.)

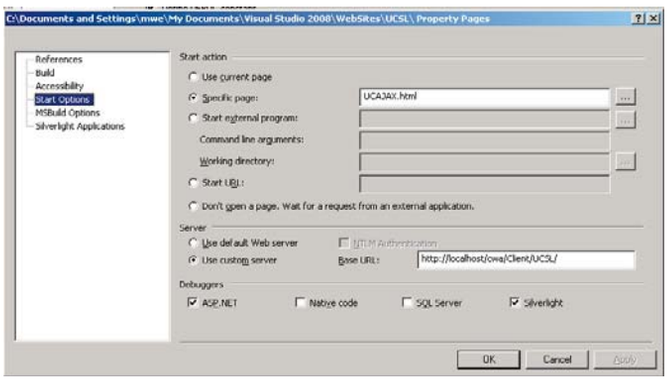
```
private ScriptObject createHttpRequest()
        {
            if (HtmlPage.Window.GetProperty("XMLHttpRequest")
== null)
            {
               return HtmlPage.Window.
CreateInstance("ActiveXObject", "Msxml2.XMLHTTP.3.0");
            }
            return HtmlPage.Window.CreateInstance("XMLHttpReq
uest");
        }
```

CODEVOORBEELD 1

Voor de verificatie en het versturen van commando's naar de CWA-server hebben we dit object nodig. Alle andere communicatie kunnen we uitvoeren met het Silverlight System.Net.HttpWebRequest-object. Dit ondersteunt asynchrone communicatie, wat vooral erg belangrijk is voor de UI-afhandeling. Synchrone communicatie in de UI thread kan je client blokkeren. En het XMLHttpObject draait altijd in de UI thread. Codevoorbeeld 2 beschrijft het inloggen.



AFBEELDING 1



AFBEELDING 2

```
public void SignIn(string username, string password, string domain)
{
  string LogonRequest =
<cwaRequests><logon><user>{0}\\{1}</user><password>{2}</pass-
word></logon></cwaRequests>";
              string LogonFormsUrl = _serverUrl + "/forms/Logon.
html";
                  string stringRequest = String.
Format(LogonRequest, domain, username, password);
                  SendRequest(LogonFormsUrl, stringRequest);
                  InitiateSession();
        }
private void SendRequest(string url, string stringRequest)
        {
            _httpRequest.Invoke("open", "POST", url, /* async */
false);
            _httpRequest.Invoke("setRequestHeader", "CWA-Ticket",
_authTicket);
            _httpRequest.Invoke("setRequestHeader", "Content-Ty-
pe", "application/x-www-form-urlencoded");
            _httpRequest.Invoke("send", stringRequest);
            string ticket = _httpRequest.
Invoke("getResponseHeader", "CWA-Ticket").ToString();
            if (ticket != "")
            {
                _authTicket = ticket;
            }
            if (_httpRequest.GetProperty("readyState").ToString()
== "4"
                && _httpRequest.GetProperty("status").ToString()
== "200" && _authTicket != "")
            {
                _isConnected = true;
                string responseXml = (string)_httpRequest.
GetProperty("responseText");
                Stream responseStream = new MemoryStream(Encoding.
UTF8.GetBytes(responseXml));
                processResponse(responseStream);
                responseStream.Dispose();
            }
        }
```

CODEVOORBEELD 2

Versturen van commando's

Codevoorbeeld 2 haalt het CWA-ticket uit de response header. In de response body zit nog meer informatie om de sessie met de CWA-server op te zetten. Het initiëren van de sessie met de CWA-server is na het aanloggen het eerste aan de server gestuurde commando. Pas na het opzetten van de sessie is het mogelijk Instant Messaging (IM)-sessies op te starten en de online-status van anderen op te vragen. Het session ID is nodig bij alle verdere communicatie binnen de sessie (codevoorbeeld 3).

```
private void processResponse(Stream responseBody)
        {
            XmlReader reader = XmlReader.Create(responseBody);
            while (reader.Read())
            {
                switch (reader.LocalName.ToLower())
                {
                    case "signInData":
                        _signInData = reader.ReadInnerXml();
                        _isConnected = true;
                        break;
                    case "uri":
                        _uri = reader.ReadInnerXml();
                        break;
                    case "PollWaitTime":
                        _pollWaitTime = Convert.ToInt32(reader.
ReadInnerXml());
                        break;
                    case "sid":
                        _sessionId = Convert.ToInt32(reader.Read-
InnerXml());
                        SignedIn.Invoke(this, new
SignInEventArgs("Succeeded", true));
                        break;
                    case "error":
                        string error = reader.ReadOuterXml();
                        break;
                }
            }
            reader.Close();
        }
private string CommandChannelUrl
        {
            get { return this._serverUrl + "/cwa/MainCommandHand-
ler.ashx"; }
        }
public void InitiateSession()
        {
            string InitiateSessionRequest = "<cwaRe-
quests xmlns=\"http://schemas.microsoft.com/2006/09/rtc/
cwa\"><initiateSession rid=\"1\">" +
            "<securityMode>private</securityMode>" +
            "<options autoPublishMachineState=\"true\"
autoSubscribePresenceForContacts=\"true\"/>" +
            "<categoryEventFilter></categoryEventFilter>" +
            "<signInData>{0}</signInData>" +
            "</initiateSession></cwaRequests>";
            string stringRequest = String.
Format(InitiateSessionRequest, _signInData);
            SendRequest(CommandChannelUrl, stringRequest);
        }
```

CODEVOORBEELD 3

Het versturen van andere commando's gaat op dezelfde manier. Het enige antwoord dat de CWA-server teruggeeft, is of het commando goed aangekomen is en begrepen wordt door de CWA-server.

Pollen van events

Na het inloggen en versturen van commando's bestaat de volgende stap uit het pollen naar informatie die van de CWA-server terugkomt. De server geeft zelf aan hoe snel er naar nieuwe informatie gevraagd mag worden. Dit ligt tussen de één en drie seconden en hangt af van de hoeveelheid activiteiten op de server. Om zeker te weten dat er geen informatie verloren gaat, wordt elke keer het ID van de laatst ontvangen informatie meegezonden. Belangrijk is geen request te sturen zolang er nog een request onderweg is (codevoorbeeld 4)

```
private string DataChannelUrl
        {
            get
            {
                return this._serverUrl +
                    "/cwa/asyncdatachannel.ashx?Sid=" + _sessio-
nId +
                    "&AckID=" + this._ackId +
                    "&UA=true";
            }
        }
```

```
public void CheckEvents()
        {
            if (_request == null)
            {
                Uri uri = new Uri(DataChannelUrl);
                _request = (HttpWebRequest)HttpWebRequest.
Create(uri);
                _request.Method = "GET";
                _request.Headers["CWA-Ticket"] = _authTicket;
                _request.BeginGetResponse(new
AsyncCallback(ResponseCallback), _request);
            }
        }
private void ResponseCallback(IAsyncResult asynchronousResult)
        {
            HttpWebRequest request = (HttpWebRequest)asynchronous-
Result.AsyncState;
            HttpWebResponse response = (HttpWebResponse)request.
EndGetResponse(asynchronousResult);
            Stream responseStream = response.GetResponseStream();
            processDataResponse(responseStream);
             responseStream.Dispose();
             _request = null;
        }
```

CODEVOORBEELD 4

In de reponse stream zit alle informatie die van de CWA-server terugkomt. Belangrijk is om het bevestiging-ID (ackid) bij het volgende verzoek mee te sturen. Verschillende events sturen informatie terug. In de CWA AJAX sdk staat een uitgebreide beschrijving van alle events en terugontvangen informatie (codevoorbeeld 5).

```
private void processDataResponse(Stream responseBody)
        {
            XmlReader reader = XmlReader.Create(responseBody);
            int rid = 0;
            while (reader.Read())
            {
                switch (reader.LocalName.ToLower())
                {
                    case "cwaevents":
                        string ackIdAttribute = reader.
GetAttribute("ackId");
                        if (ackIdAttribute != null && ackIdAttri-
bute != string.Empty)
                        {
                            _ackId = int.Parse(ackIdAttribute);
                        }
                        break;
                    case "conference":
                        break;
                    case "contactgroup":
                        ProcessContactGroup(reader.ReadOuter-
Xml());
                        break;
                    case "userpresence":
                        break;
                    case "selfpresence":
                        break;
                    case "subscribers":
                        break;
                    case "searchresult":
                        break;
                    case "pollfailed":
                        break;
                    default:
                        break;
                }
            }
            reader.Close();
        }
```

CODEVOORBEELD 5

De volgende stap bestaat uit het agenderen van het pollen. Hiervoor gebruiken we een Storyboard van Silverlight (codevoorbeeld 6).

```
<UserControl.Resources>
       <Storyboard x:Name="Timer" Completed="TimerCompleted"/>
</UserControl.Resources>
```

CODEVOORBEELD 6

In de bijbehorende code file, voegen we nu de code toe die het TimerCompleted event afhandelt. Acties in deze methode: zet de tijdsduur, stop de timer en kijk of er nieuwe events zijn, start daarna de timer weer. De eerste keer moet je na het inloggen de timer handmatig starten (codevoorbeeld 7).

```
private void TimerCompleted(object sender, EventArgs e)
        {
            Timer.Duration = new TimeSpan(0,0,0,0,_userAgent.Pol-
lWaitTime );
            Timer.Stop();
            _userAgent.CheckEvents();
            Timer.Begin();
        }
```

CODEVOORBEELD 7

Dit is de basis voor het maken van een CWA Silverlight Client.

User Interface

Voeg in de Page.xaml een StackPanel toe met de naam Contacts. In dit StackPanel komen de groepen en contactpersonen (codevoorbeelden 8 en 9).

```
<UserControl x:Class="UCAJAX.ContactControl"
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d=http://schemas.microsoft.com/expression/blend/2008
    xmlns:mc=http://schemas.openxmlformats.org/markup-compatibi-
lity/2006
    xmlns:c="clr-namespace:System.Windows.
Controls;assembly=System.Windows.Controls"
    mc:Ignorable="d"
   d:DesignWidth="640" d:DesignHeight="480">
    <Grid x:Name="LayoutRoot" Background="White" Height="18" >
       <Ellipse x:Name="status" Height="14"
HorizontalAlignment="Left" Margin="10,1,0,0"
VerticalAlignment="Top" Width="15" Stroke="Wheat" RenderTransfor-
mOrigin= "-2.29999995231628,-2.09999990463257">
       <Ellipse.Fill>
          <RadialGradientBrush>
            <GradientStop Color="#FFFFFFFF" Offset="0.067"/>
            <GradientStop x:Name="statuscolor"  Color="Wheat" Off-
set="0.951"/>
                              </RadialGradientBrush>
       </Ellipse.Fill>
     </Ellipse>
     <TextBlock  x:Name="text" Height="15"
HorizontalAlignment="Left" Width="185" Margin="28,-1,0,0"
VerticalAlignment="Top" Text="" FontSize="12" />
      <TextBlock  x:Name="statustext" Height="15"
HorizontalAlignment="Left" Width="80" Margin="185,0,0,0"
VerticalAlignment="Top" Text="" FontSize="12"/>
    </Grid>
</UserControl>

In de code file komt het volgende:
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace UCAJAX
{
```

```
    public partial class ContactControl : UserControl
    {
        private Contact _contact;
        public ContactControl()
        {
            InitializeComponent();
        }
        public Contact contact
        {
            get { return _contact; }
            set
            {
                _contact = value;
                if (_contact.DisplayName != null)
                {
                    this.text.Text = _contact.DisplayName;
                }
                else
                {
                    this.text.Text = _contact.Uri.Replace("sip:",
"");
                }
            }
        }
}
```

CODEVOORBEELD 8. HET CONTACTS-CONTROL

```
<UserControl
  xmlns=http://schemas.microsoft.com/client/2007
  xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
  xmlns:d=http://schemas.microsoft.com/expression/blend/2008
  xmlns:mc=http://schemas.openxmlformats.org/markup-compatibili-
ty/2006
  mc:Ignorable="d"
  x:Class="UCAJAX.GroupControl"
  d:DesignWidth="640" d:DesignHeight="480" xmlns:UCAJAX="clr-
namespace:UCAJAX">
  <Grid x:Name="LayoutRoot" Background="White" >
    <Grid HorizontalAlignment="Stretch" VerticalAlignment="Top"
Height="20" MouseLeftButtonDown="expandCollapse" Cursor="Hand">
      <TextBlock  x:Name="icon" Height="20"
HorizontalAlignment="Left" Margin="0,2,0,0"
VerticalAlignment="Top" Text="4" Width="20" FontFamily="Webdings"
/>
      <TextBlock  x:Name="text" Height="20"
HorizontalAlignment="Stretch" Margin="20,0,0,0"
VerticalAlignment="Top" Text="" FontWeight="Bold"
d:IsHidden="True" FontSize="12"/>
    </Grid>
      <Grid x:Name="Contactsgrid" HorizontalAlignment="Stretch"
VerticalAlignment="Top" Margin="0,20,00,0" Visibility="Collapsed">
          <StackPanel x:Name="Contacts"
HorizontalAlignment="Left" Margin="0,0,0,0"
VerticalAlignment="Stretch" Width="270"/>
    </Grid>
  </Grid>
</UserControl>
In de code file komt het volgende:
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace UCAJAX
{
  public partial class GroupControl : UserControl
  {
    public GroupControl()
    {
      InitializeComponent();
    }
      private void expandCollapse(object sender, MouseButton-
EventArgs e)
      {
          if (Contactsgrid.Visibility == Visibility.Collapsed)
          {
              this.icon.Text = "6";
              Contactsgrid.Visibility = Visibility.Visible;
          }
          else
          {
              this.icon.Text = "4";
              Contactsgrid.Visibility = Visibility.Collapsed;
          }
      }
    }
}
```

CODEVOORBEELD 9. HET GROEPCONTROL

De twee controls in codevoorbeelden 8 en 9 vormen een basis voor het tonen van groepen en de bijbehorende contactpersonen. Na het aanloggen op de CWA-server komt er een aantal events binnen, zoals het contactgroup event. De gegevens van dit event gaan in een EventsArg-object. Het object raised een GroupEvent als er een nieuwe groep binnenkomt en een ContactEvent als er een nieuw contact binnenkomt (codevoorbeeld 10).

```
private void ProcessContactGroup(string xml)
        {
            Stream xmlStream = new MemoryStream(Encoding.UTF8.
GetBytes(xml));
            XmlReader reader = XmlReader.Create(xmlStream);
           while (reader.Read())
            {
                switch (reader.LocalName.ToLower())
                {
                    case "group":
                        GroupEventArgs geventArgs = new
GroupEventArgs(reader.ReadOuterXml());
                        if (GroupEvent != null)
                        {
                            GroupEvent.Invoke(this, geventArgs);
                        }
                        break;
                    case "contact":
                       ContactEventArgs ceventArgs = new
ContactEventArgs(reader.ReadOuterXml());
                        if (ContactEvent != null)
                        {
                            ContactEvent.Invoke(this, ceventArgs);
                        }
                        break;
                }
            }
        }
```

CODEVOORBEELD 10

Om de gegevens bij te houden van de groepen en contactpersonen zijn een dictionary met de groeps- en contacts-gegevens nodig (codevoorbeeld 11)

```
public class Contact
    {
        public string Uri;
        public string Action;
        public string AccessLevel;
        public string DisplayName;
        public string Note;
        public string Company;
        public string Title;
        public string Office;
        public string[] Groups;
        public bool Subscribed;
        public bool Taged;
        public int Availability;
    }
public class Group
    {
        public int Id;
        public string Action;
```

Wij zoeken nieuwe no-nonsense collega's, mét gevoel voor humor.

Cylogy selecteert, ontwikkelt, implementeert en beheert Content Management oplossingen waarmee organisaties effectiever kunnen communiceren. We werken aan uitdagende projecten voor aansprekende klanten, met gebruik van de modernste technologie. Cylogy kent een open en integere cultuur waar plezier en samenwerken centraal staan. Geen 'lagen', geen dichte deuren en geen zware procedures. Iedereen is gelijk, en eigen verantwoordelijkheid nemen telt meer dan strak de regeltjes volgen.

Omdat onze klanten continu nieuwe projecten starten, zoeken wij Content Management Consultants en Specialisten met kennis van Microsoft (Sharepoint Server, Commerce Server, SQL Server, BizTalk Server), Tridion en Smartsite. Echte Experts dus.

Heb je ervaring met één of meerdere van deze producten en wil je deel uitmaken van een team van professionals die helemaal gek zijn van hun vak, aarzel dan niet en help ons nóg beter te worden.

Klaar om je carrière een boost te geven? Bekijk dan alle vacatures op onze website.

CYLOGY
Content Management
Solutions & Specialists

Cylogy Nederland B.V.
Atlas Complex, Hoogoorddreef 9, 1101 BA Amsterdam
T: +31 (0) 20 - 312 0 636 | F: +31 (0) 20 - 312 0 444
www.cylogy.nl | E-mail: careers@cylogy.nl

```
        public string Name;
    }
In de code bij de Page.xaml staan deze dictionaries.

private Dictionary<int, Group> _groupList = new Dictionary<int,
Group>();
private Dictionary<string, Contact> _contactList = new
Dictionary<string, Contact>();
In de page.xaml code worden de events afgehandeld die geraised
worden.
_userAgent = new UserAgent();
 _userAgent.SignIn(userName.Text, passWord.Text, domain.Text);
 _userAgent.GroupEvent += new EventHandler<GroupEventArgs>(_user-
Agent_GroupEvent);
 _userAgent.ContactEvent += new EventHandler<ContactEventArgs>(_
userAgent_ContactEvent);
```

CODEVOORBEELD 11

Er kunnen verschillende acties binnenkomen voor het toevoegen verwijderen en hernoemen. Codevoorbeeld 12 laat alleen het toevoegen van groepen en contacts zien.

```
void _userAgent_GroupEvent(object sender, GroupEventArgs e)
        {
            Dispatcher.BeginInvoke(delegate()
           {
            try
            {
                switch (e.Group.Action)
                {
                    case "added":
                        _groupList.Add(e.Group.Id, e.Group);
                        AddContactGroup(e.Group);
                        if (e.Group.Name == "~")
                        {
                            _reservedGroupId = e.Group.
Id.ToString();
                        }
                        break;
                    case "updated":
                        break;
                    case "deleted":
                        break;
                    default:
                        break;
                }
            }
            catch (Exception ex)
            {
                //TODO: Add Exception Handle code here
            }
  });
}
```

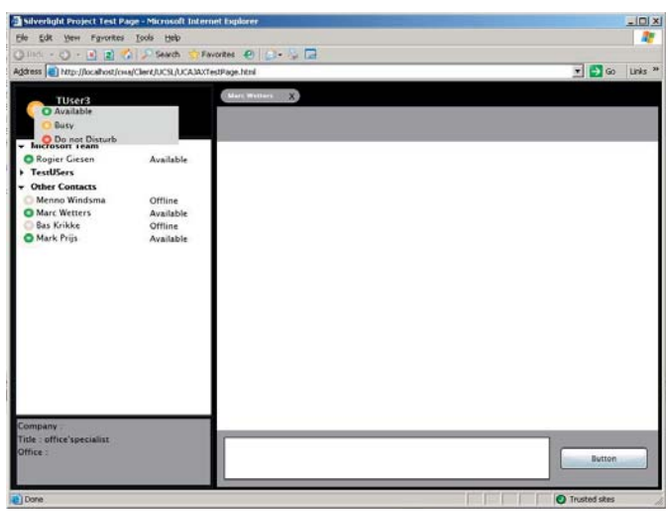
CODEVOORBEELD 12

De gegevens over de groep worden in de dictionary en in de UI toegevoegd (codevoorbeeld 13).

```
private void AddContactGroup(Group group)
        {
            if (group.Name != "~")
            {
                GroupControl ctrl = new GroupControl();
                ctrl.Tag = "group" + group.Id.ToString();
                ctrl.text.Text = group.Name;
                this.Contacts.Children.Insert(0,ctrl);
            }
        }

void _userAgent_ContactEvent(object sender, ContactEventArgs e)
 {
    Dispatcher.BeginInvoke(delegate()
   {
      switch (e.Contact.Action)
      {
          case "added":
              if (_contactList.ContainsKey(e.Contact.Uri) ==
false)
              {
                  _contactList.Add(e.Contact.Uri, e.Contact);
                  AddContact(_contactList[e.Contact.Uri]);
              }
              break;
          case "updated":
              break;
          case "deleted":
              break;
          default:
              break;
      }
  });
 }
```

CODEVOORBEELD 13



AFBEELDING 3

Een contactpersoon kan in meer groepen staan en moet dus bij elke groep getoond worden. Naast de door de gebruiker aangemaakte groepen is er een standaard groep met daarin alle contactpersonen. In UI worden in die groep "overige contact personen" alleen die contactpersonen getoond, die niet in een andere groep zitten (codevoorbeeld 14)

```
private void AddContact(Contact contact)
        {
            bool added = false;
                if (contact.Groups != null && contact.Groups.
Length >= 1)
                {
                    foreach (string groupId in contact.Groups)
                    {
                        if (groupId != _reservedGroupId)
                        {
                            GroupControl grpctrl =
GetGroupControlByTag("group" + groupId);
                            if (grpctrl != null)
                            {
                            ContactControl ctrl = new Contact-
Control();
                            ctrl.contact = contact;
                            ctrl.Tag = "grp" + groupId + "contact"
+ contact.Uri;
                            grpctrl.Contacts.Children.Add(ctrl);
                            added = true;
                            }
                        }
                    }
                }
```

```
                if (!added)
                {
                    GroupControl grpctrl = GetGroupControlByTag
("groupothercontacts");
                    if (grpctrl != null)
                    {
                        ContactControl ctrl = new ContactCon-
trol();
                        ctrl.contact = contact;
                        ctrl.Tag = "grpgroupothercontactscontact"
+ contact.Uri;
                        grpctrl.Contacts.Children.Add(ctrl);
                        added = true;
                    }
                }
        }
```

CODEVOORBEELD 14

We moeten er nu alleen nog op letten dat alle callbacks van het HttpWebRequest niet in de UI thread terugkomen! Om met deze gegevens toch de UI te updaten, maken we gebruik van het Dispatcher-object (codevoorbeeld 15).

```
Dispatcher.BeginInvoke(delegate()
  {
…. Update UI
})
```

CODEVOORBEELD 15

In de events die van de CWA-server terugkomen, staan alleen de gewijzigde gegevens. Op het moment dat bijvoorbeeld de display namen van een contactpersoon wijzigt, worden niet alle gegevens van deze contactpersoon meegestuurd, alleen de gewijzigde. Op dezelfde manier kunnen we andere events afhandelen en in de UI laten zien, zoals presence-statuswijzigingen en berichten.

Links

http://unified-communications-development.blogspot.com

AJAX sdk

http://www.microsoft.com/downloads/details.aspx?FamilyID=d5a36cc7-9b94-4082-ab55-22feffce6b80&DisplayLang=en

Marc Wetters is werkzaam bij e-office. E-mail: Marc.Wetters@e-office.com.