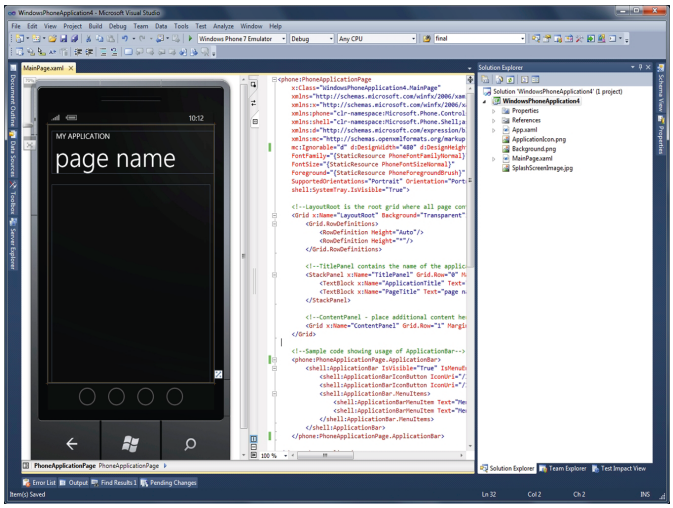
Windows Phone 7 en het Azure Platform

Krachtige combinatie van verschillende technologieën

Dennie Bastiaan

Het waren twee hoofditems op de PDC van 2010: Windows Phone 7 en Windows Azure. Azure, de clouddient van Microsoft, die in het begin van 2010 werd vrijgegeven voor het grote publiek. Daarnaast is dit jaar Windows Phone 7 voorgesteld aan ontwikkelaars en zijn sinds oktober de eerste telefoons ermee beschikbaar. Het zijn twee van elkaar losstaande technologieën die op het eerste gezicht niets met elkaar te maken hebben. Echter, niets is minder waar.

Microsoft heeft voor Windows Phone 7 een belangrijke beslissing genomen: Silverlight en XNA zijn dé applicatieplatformen voor Windows Phone 7. Applicaties moeten met C# en Silverlight of XNA worden ontwikkeld. In de toekomst zal VB.NET aan dat rijtje worden toegevoegd, maar belangrijk is dat .NET technologie een prominente rol heeft gekregen.



FIGUUR 1.

Wat betekent dit voor ontwikkelaars? Iedere .NET-ontwikkelaar is eigenlijk al een Windows Phone 7 ontwikkelaar. De ontwikkelaar maakt echter geen applicaties die in een browser draaien, maar op een telefoon. Technisch en syntactisch gezien is er vrij weinig verschil. Het is nog steeds een applicatie, draaiend op hardware, gebouwd met .NET en C#. Het verschil zit vooral in de manier waarop software wordt ontwikkeld en welke concepten er worden gebruikt. Een browser of een desktop is nu eenmaal geen smartphone. Wat maakt een smartphone dan zo anders? Dit artikel legt uit op welke kenmerken je moet letten en hoe je deze kenmerken kunt implementeren.

Opkomst van mobiel internet

Het zal niemand zijn ontgaan dat het internet de laatste jaren een transitie heeft gemaakt van de aansluiting aan huis naar de draadloze mobiele aansluiting. Notebooks, mobiele telefoons en zelfs auto's hebben tegenwoordig een aansluiting op het internet. Telecomproviders hebben het gebruik van mobiel internet de afgelopen jaren explosief zien stijgen en de verwachting is dat het verbruik de komende jaren zal verdrievoudigen. Dit is dus niet zomaar een hype, maar een trend die goed zal doorzetten.
Voor software-ontwikkelaars heeft deze trend gevolgen die belangrijk zijn bij het ontwikkelen van mobiele applicaties. Er zijn meer mogelijkheden om uit te kiezen en dus meer valkuilen. Om optimaal gebruik te maken van deze trend zal een ontwikkelaar moeten stilstaan bij een aantal eigenschappen van hedendaagse mobiele applicaties.
Allereerst valt op dat applicaties 'connected' zijn. Via het internet communiceren de applicaties met elkaar. Denk bijvoorbeeld aan online games tegen elkaar spelen, online brainstormen met een collaboratieve mindmap of simpelweg beeldbellen.
Een ander kenmerk van mobiele applicaties is 'data snacking'. De applicaties kunnen via online services op de hoogte worden gehouden van realtime informatie. Bijvoorbeeld een routeplanner die het omzeilen van files in de route verwerkt of een datumplanner voor groepsuitjes die alternatieve datums voorstelt op basis van weersinformatie.
Nog een kenmerk is het gebruik van online opslag. Steeds meer informatie zal online worden opgeslagen als vervanging voor het gebruik van lokale opslag. Denk hierbij aan notities, contacten, foto's, filmpjes, documenten en dergelijke. Online opslag van informatie kent veel voordelen. Online staat je informatie veilig. De informatie is vanaf meerdere plekken beschikbaar en je informatie is gemakkelijk deelbaar. Bijvoorbeeld een fotoalbum dat je wilt delen met je familie of vrienden.

Moderne mobiele applicaties en Azure

Wat hebben deze kenmerken van moderne mobiele applicaties te

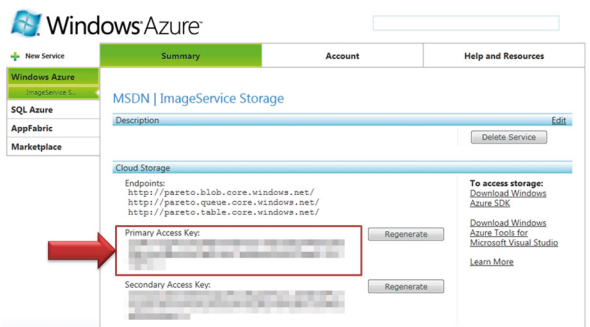
maken met Windows Phone 7 en Azure? Alles! Azure kent namelijk een aantal diensten, dat direct aansluit op de drie bovengenoemde eigenschappen.

FIGUUR 2.

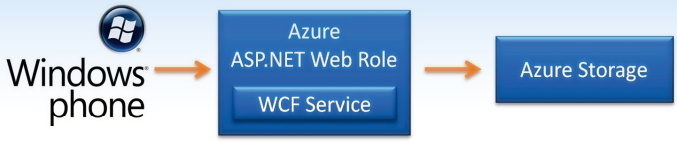
De Azure AppFabric Service Bus voorziet in de mogelijkheid connected applicaties te ontwikkelen. De Service Bus maakt bidirectionele communicatie tussen applicaties mogelijk op een veilige manier, zonder dat er rekening gehouden hoeft te worden met firewalls of de grenzen van het netwerk. De Service Bus zorgt ervoor dat een ontwikkelaar niet met de verbinding zelf bezig is, maar met datgene waar de verbinding voor bedoeld is. Dezelfde technologie wordt bijvoorbeeld ook gebruikt in applicaties als Windows Live Messenger.
SQL Azure en OData voorzien in de mogelijkheid in een applicatie te 'data snacken'. OData is een lichtgewicht protocol dat toegang geeft tot de gegevenssets. Het systeem ondersteunt het ophalen en aanpassen van gegevens. De gegevens staan vervolgens in een SQL Server database in SQL Azure. OData kan in combinatie met LINQ worden gebruikt en hoeft dus voor de ontwikkelaar niet moeilijk te zijn.
Azure Storage voorziet in de opslag van gegevens en bestanden in je applicatie. Het maakt het mogelijk om informatie in de Cloud op te slaan. Storage ondersteunt een aantal scenario's, waaronder een message queue en blob storage voor het opslaan van bestanden.
Azure biedt dus een scala aan clouddiensten die essentieel zijn bij het bouwen van moderne mobiele applicaties. Veel diensten zijn gebaseerd op open standaarden en zijn ook beschikbaar voor andere platformen dan .NET alleen. De integratie met .NET en Silverlight maken het echter een platform waar Windows Phone ontwikkelaars veel mee zullen werken. Alle diensten van Azure zijn geïntegreerd in of aanspreekbaar met .NET (zoals WCF en LINQ) en er is een SDK beschikbaar (de Azure SDK) die communicatie met Azure diensten vergemakkelijkt.

Voorbeeld: Windows Phone en Cloudstorage

We maken een voorbeeldapplicatie die plaatjes kan opslaan in en lezen uit de Cloud. De gegevens (in dit geval een plaatje) moeten



FIGUUR 3.



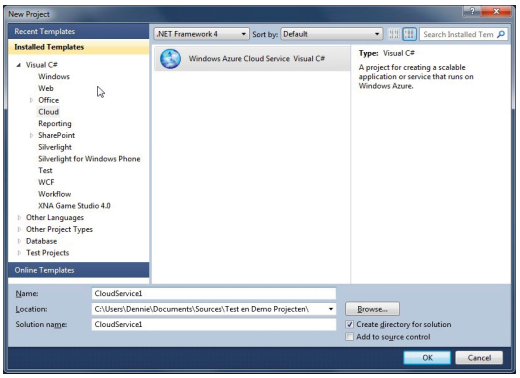
FIGUUR 4.

worden opgeslagen in Azure Storage. Hiervoor kan de Azure SDK worden gebruikt. De applicatie maakt een connectie met Azure en plaatst vervolgens een blob in een zogenaamde blobcontainer. De connectie wordt gemaakt op basis van een Cloud Storage Account. Dit zijn de credentials en een URI van de storage. Onderdeel hiervan is de 'Primary Access Key'. Deze sleutel is nodig om te mogen communiceren met Azure Storage.

Er is echter een klassiek probleem met deze sleutel. Om gebruik te maken van deze dienst in de applicatie, moeten deze credentials inclusief de sleutel mee worden geleverd met de applicatie. Dit terwijl de sleutel strikt persoonlijk is en niet gedeeld mag worden. Bij een potentieel succesvolle App in de Marketplace zou de sleutel over de hele wereld in telefoons terechtkomen.

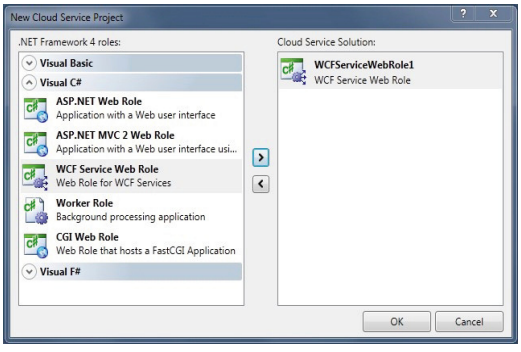
WCF Service Web Role

De oplossing is om een WCF service te hosten in Windows Azure. Deze WCF service kan dan volledig custom worden ontwikkeld met de functionaliteiten die nodig zijn voor de applicatie, waaronder authenticatie voor gebruikers van de Windows Phone app. De Azure SDK (deze dient geïnstalleerd te zijn) kent het zogenaamde Windows Azure Cloud Service projecttype.



FIGUUR 5.

Zodra een dergelijk project wordt gecreëerd, wordt gevraagd wat de rol van de cloudservice zal zijn. In dit geval zijn we geïnteresseerd in de 'WCF Service Web Role'. Deze web role zorgt voor een WCF Service met daarbij een project voor de configuratie van



FIGUUR 6.

de Azure Web Role. Deze werkwijze heeft een groot voordeel. Je kunt de service lokaal testen in Visual Studio en zodra een release gereed is, kan deze vanuit Visual Studio worden gedeployed naar Azure.

In codevoorbeeld 1 staat het simpele contract van de voorbeeldservice. We noemen deze service de ImageService. We ondersteunen twee simpele operaties. Het opslaan van een plaatje en het ophalen van de lijst met plaatjes. Het ImageItem object is een DataContract met daarin een Name, een Date en een Image als byte array. Ook staat in dit codevoorbeeld de implementatie van SaveImageItem. SaveImageItem pusht het plaatje in de storage.

```
[ServiceContract]
public interface IImageService
{
    [OperationContract]
    void SaveImageItem(ImageItem imageItem);

    [OperationContract]
    IEnumerable<ImageItem> ListImageItems();
}

public void SaveImageItem(ImageItem imageItem)
{
    CloudStorageAccount account = CloudStorageAccount.
DevelopmentStorageAccount;
    CloudBlobClient client = account.CreateCloudBlobClient();
    CloudBlobContainer container = client.GetContainerReference
("imagecontainer");

    CloudBlob blob = container.GetBlobReference(Guid.NewGuid().
ToString());

    BlobStream blobStream = blob.OpenWrite();
    blobStream.Write(imageItem.Image, 0, imageItem.Image.Count());
    blobStream.Close();

    blob.Metadata["Name"] = imageItem.Name;
    blob.Metadata["Date"] = imageItem.Date;
    blob.SetMetadata();
}
```

CODEVOORBEELD 1.

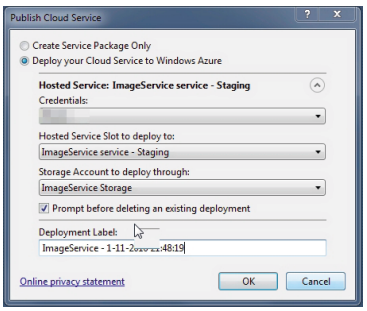
Zoals te zien is in codevoorbeeld 1 wordt een CloudStorageAccount aangemaakt. De CloudBlobClient is een proxy die toegang geeft tot de service. Vervolgens openen we een blob container. Een blob container kan worden vergeleken met een map op een file system. Deze container is een plek waar blobs bij elkaar staan. Met container.GetBlobReference(...) halen we een enkele blob uit de storage. Als deze niet bestaat, dan wordt deze aangemaakt. We vullen de blob met een byte array met behulp van een BlobStream. Daarna stellen we de metadata van de blob in. Belangrijk is dat de metadata pas wordt ingesteld nadat de blob is gemaakt en gevuld is met data. Stel je de metadata in voordat er een blob is weggeschreven, dan zal de metadata verloren gaan. Codevoorbeeld 2 toont de implementatie van ListImageItems(). Met 'ListBlobs' worden alle blobs die in een blob container aanwezig zijn uit de storage opgehaald. In dit geval wordt gebruik gemaakt van één enkele blob container.

```
public IEnumerable<ImageItem> ListImageItems()
{
    List<ImageItem> imageItems = new List<ImageItem>();

    CloudStorageAccount account = CloudStorageAccount.
DevelopmentStorageAccount;
    CloudBlobClient client = account.CreateCloudBlobClient();
    CloudBlobContainer container = client.GetContainerReference
("imagecontainer");
    IEnumerable<IListBlobItem> blobList = container.ListBlobs();
    foreach (var item in blobList)
    {
        CloudBlob blob = container.GetBlobReference
(item.Uri.ToString());
        BlobStream blobStream = blob.OpenRead();
        blob.FetchAttributes();

        byte[] buffer = new byte[blobStream.Length];
        blobStream.Read(buffer, 0, Convert.ToInt32(blobStream.
Length));
        blobStream.Close();

        imageItems.Add(new ImageItem()
        {
            ID = item.Uri.ToString(),
            Name = blob.Metadata["Name"],
            Date = blob.Metadata["Date"],
            Image = buffer
        });
    }

    return imageItems;
}
```

CODEVOORBEELD 2.

Service Deployment

Zodra een service gereed is om released te worden, kan de service via Visual Studio worden gedeployed naar Azure. Onder het contextmenu van het Cloud Service project staat een optie 'Publish'. Als deze optie wordt gekozen volgt een scherm waarin de properties van de deployment kunnen worden ingesteld.



FIGUUR 7.

Er is een aantal voorwaarden voordat deze service gedeployed kan worden: Er moet een API Certificate zijn gemaakt en geüpload naar Azure. Deze kunnen we maken in Visual Studio door bij 'Credentials' voor 'Add' te kiezen. Daarnaast moet er in Windows Azure een service slot zijn aangemaakt. Zodra de credentials in orde zijn, haalt het Publish scherm alle service slots op. Als we op OK klikken, zal Visual Studio de service automatisch deployen naar Azure. Via Staging kan in de developer portal van Azure worden besloten of de service in productie kan gaan. Belangrijk is wel dat er een gedegen serviceconfiguratie is (ServiceConfiguration.cscfg). De standaardconfiguratie van het Cloud Service Project is een local configuratie die niet zomaar zal werken in Azure. Op MSDN Library is meer te vinden over hoe je een serviceconfiguratie kunt maken. Zie http://msdn.microsoft.com/en-us/library/dd179398.aspx.

De Windows Phone Applicatie

Als de service gereed is, beginnen we aan de implementatie van de Windows Phone 7 app. Het bouwen aan Windows Phone 7

applicaties verschilt niet veel van het ontwikkelen van een gewone Silverlight applicatie. Een formulier wordt een 'PhoneApplicationPage' genoemd. Een page ondersteunt XAML. Grids, StackPanels en de gebruikelijke controls zoals een TextBox of een Button zijn onderdeel van de toolbox van Windows Phone 7.
Zodra er een PhoneApplicationPage is, plaatsten we een ListBox om de data weer te geven:

```
<ListBox x:Name="ImageList">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal" Margin="3">
                <Image Source="{Binding AnImage}" Width="150" />
                <StackPanel Orientation="Vertical" Width="300">
                    <TextBlock Text="{Binding Name}"
FontSize="24" FontWeight="Bold" Width="295" />
                    <TextBlock Text="{Binding Date}"
FontSize="18" Width="295" />
                </StackPanel>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

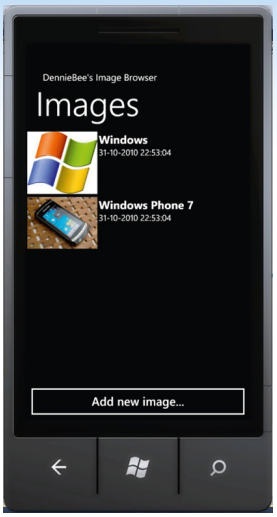
CODEVOORBEELD 3.

De ListBox 'ImageList' vullen we in de code behind van deze page. Elke PhoneApplicationPage kent een overridable methode die 'OnNavigatedTo' heet. Dit betekent dat de code hierin altijd wordt uitgevoerd, zodra de gebruiker op de pagina terechtkomt. In dit geval instantiëren we een ImageServiceClient service proxy van de service. We bevragen de service asynchroon, zodat de UI thread gespaard blijft bij het ophalen van de gegevens. Zodra de gegevens zijn ontvangen worden ze gebind aan de ImageList.

```
ImageServiceClient client;

protected override void OnNavigatedTo(System.Windows.Navigation.
NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    client = new AzureServices.ImageServiceClient();
    client.ListImageItemsCompleted +=
        new EventHandler<ListImageItemsCompletedEventArgs>(
            client_ListImageItemsCompleted);
    client.ListImageItemsAsync();
}

void client_ListImageItemsCompleted(object sender, AzureServices.
ListImageItemsCompletedEventArgs e)
{
    var source =
        from item in e.Result
        select new Item()
        {
            ID = item.ID,
            Name = String.IsNullOrEmpty(item.Name) ? "-" : item.
Name,
            Date = String.IsNullOrEmpty(item.Date) ? "-" : item.
Date,
            AnImage = GetImageFromBytes(item.Image)
        };

    ImageList.ItemsSource = source;
}
```

CODEVOORBEELD 4.

Het aanspreken van SaveImageItem verloopt op dezelfde wijze. Zo kan vanuit de telefoon een plaatje naar onze storage worden gepusht.



FIGUUR 8.

Conclusie

Dit artikel beschrijft hoe je Azure Storage kunt gebruiken in je applicaties. Dit is echter slechts één van de vele mogelijkheden die Azure in combinatie met Windows Phone biedt. De mogelijkheden lijken eindeloos. Wie het interessant vindt, kan zich verdiepen in onderwerpen als de AppFrabric Service Bus, SQL Azure met OData en het schrijven van bijvoorbeeld een Worker Role als CPU-intensief onderdeel van je applicatie zaken om te onderzoeken.
Windows Phone 7 is een telefoon waarbij connectiviteit een belangrijke rol speelt. Azure zal voor een groot deel deze rol gaan invullen. Het is slechts het begin van wat komen gaat. Azure zal een steeds groter aantal scenario's gaan ondersteunen en Windows Phone 7 blijft zich vernieuwen. De tools die nu ter beschikking zijn om moderne mobiele applicaties te maken zijn echter al flink volwassen en tonen aan dat de ontwikkelaar zich kan richten op de applicatie.
Windows Azure en Windows Phone 7 lijken dus wellicht in eerste instantie niet zoveel gemeen te hebben. De realiteit is echter dat je als Windows Phone 7 developer bijna niet zonder kan indien je van plan bent om moderne mobiele applicaties te bouwen.

Bezig met een Windows Phone applicatie of van plan om er één te schrijven? Voor meer informatie over Azure en de Azure SDK kun je kijken op http://www.microsoft.com/windowsazure/.

Dennie Bastiaan, is Software Architect bij Pareto. Hij is te bereiken via email. dennie.bastiaan@pareto.nl en via twitter. http://twitter.com/denniebee.