Arthur van Adrichem is technisch consultant bij Qurius AS

Een toepassing van Custom Field Types

TAXONOMIE BINNEN OFFICE SHAREPOINT SERVER

Tijdens de realisatie van een nieuw op Microsoft Office SharePoint Server 2007 gebaseerd kennissysteem liepen we al snel tegen een aantal serieuze uitdagingen aan. Het eenvoudig classificeren van informatie en documenten aan de hand van een taxonomie is standaard niet mogelijk. In dit artikel lees je hoe we deze beperking toch hebben kunnen tackelen.

Taxonomie is de wetenschap van het classificeren van voorwerpen, levende wezens, gebeurtenissen en/of informatie op basis van een vooraf vastgelegd classificatiesysteem. Eén van de beste voorbeelden van een taxonomie is de indeling van de Zweedse arts en bioloog Carolus Linnaeus. Hij heeft het systeem bedacht dat wordt gebruikt voor het indelen van planten en dieren. Theoretisch bestaat het opzetten van een classificatiesysteem uit het opsplitsen van elementen van een groep in subgroepen die in ieder geval wederzijds uitsluitend en tevens gezamenlijk uitputtend moeten zijn. Dit principe staat ook bekend als het MECE-principe (mutually exclusive, collectively exhaustive). In de praktijk moet een dergelijk systeem eenvoudig te gebruiken en overzichtelijk zijn. Hier lag de uitdaging, want hoe voeg je zoiets toe aan SharePoint Server? Hoe zorg je ervoor dat je dit systeem bij het toevoegen van verschillende soorten informatie eenvoudig kunt aanroepen? Hoe zorg je ervoor dat de groepen binnen de toevoeg- en bewerkpagina's in een overzichtelijke boomstructuur worden getoond? Het antwoord op deze vragen is jammer genoeg niet door gebruik te maken van de standaardfunctionaliteit van Office SharePoint Server.

Wat is de oplossing?

Bij het toevoegen van een nieuwe kolom aan een SharePoint-lijst dien je naast het invullen van de naam ook een type te kiezen. Je kunt kiezen uit bijvoorbeeld Single line of text, Multiple lines of text, Choice, Number, Lookup, enzovoort. Op basis van dit type kiest SharePoint het juiste dataformaat en wordt ook het best passende control getoond op de pagina's voor het toevoegen en bewerken van een item. Je kunt hierbij denken aan een textbox, een combobox of een lijst met radiobuttons. Je zou verwachten dat in ons geval gewoon kan worden volstaan met een standaard Lookup-veld, waarbij verscheidene waarden vanuit het classificatiesysteem kunnen worden geselecteerd. Het probleem bij dit veldtype is echter dat de beschikbare waarden standaard in een selectielijst (listbox) worden getoond. Het gebruik van een selectielijst wordt bij grote aantallen categorieën erg onoverzichtelijk. We wilden daarom invloed kunnen uitoefenen op de presentatie van het veld en de groepen uit het classificatiesysteem in een boomstructuur tonen. Dit is mogelijk door gebruik te maken van een Custom Field Type. Een Custom Field Type is één van de vele uitbreidingsmogelijkheden die gebruikmaakt van het SharePoint-objectmodel. Een Custom Field Type is letterlijk een zelf gedefinieerd veldtype dat overal binnen de site met behulp van een nieuwe kolom aan een lijst kan worden toegevoegd. Een Custom Field Type kan, naast het uitoefenen van invloed op de presentatie van een veld, ook nog voor twee andere scenario's worden ingezet. In tabel 1 vind je hiervoor een handig totaaloverzicht.

Wat gaan we doen?

De oplossing voor het probleem wordt in vijf eenvoudige stappen gerealiseerd.

1. Aanmaken van een lijst voor de opslag van de categorieën binnen het classificatiesysteem en een lijst voor het opslaan van documenten en informatie.
2. Opzetten van een 'Custom Field Value Class' voor de opslag en het transport van de waarde van het veld en indien noodzakelijk validatie.
3. Creëren van een 'Custom Field Editor UserControl' voor de presentatie van het veld.
4. Aanmaken van de 'Custom Field Type Definition' om aan SharePoint duidelijk te maken dat er een extra veldtype is gemaakt.
5. Deployment.

Bij alle stappen wordt een duidelijk voorbeeld gegeven. De vol-

Scenario	Uitleg
Presentatie	Standaard wordt er binnen SharePoint bij een veld van het type text een textbox getoond. Maar wat nu als je in plaats van deze textbox een colorpicker wilt laten zien die voor een bepaalde kleur de hexadecimale waarde opzoekt? Een Custom Field Type laat je door middel van het gebruik van UserControls vrij in de presentatie van het veld.
Dataformaat	Op het moment dat een stukje informatie qua opbouw of structuur niet overeenkomt met de standaard veldtypen van SharePoint kun je ook gebruikmaken van een Custom Field Type. Bijvoorbeeld als je gegevens binnen meer kolommen wilt kunnen invoeren en/of opslaan. Denk hierbij aan telefoonnummers met een landnummer.
Validatie	Binnen SharePoint is het mogelijk te controleren of een bepaald veld is ingevuld of dat een ingevoerde hyperlink correct is. Op het moment dat je extra validatiemogelijkheden wilt hebben, moet je ook gebruikmaken van Custom Field Types. Bijvoorbeeld als je op een bankrekeningnummer de elfproef wilt loslaten.

Tabel 1. Scenario's voor inzetten Custom Field Type

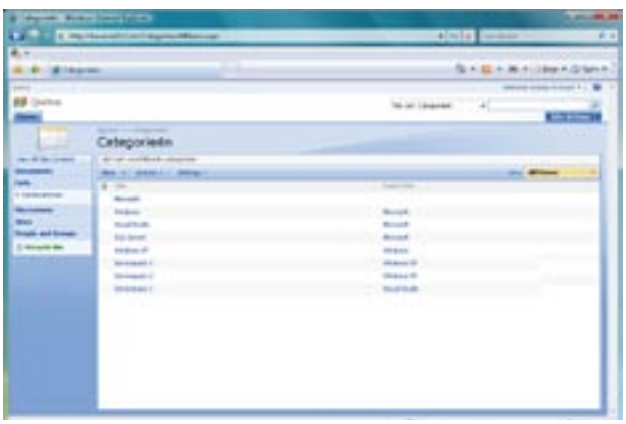
ledige code van het Custom Field Type vind je op de website van het .Net Magazine en op website van Qurius. (zie referenties).

Stap1: Aanmaken kenniscentrum en lijst met categorieën

De eerste lijst is een overzicht met daarin de verschillende groepen van het classificatiesysteem, de tweede lijst is een centrale opslagplaats voor informatie en documenten ten behoeve van de kennisdeling. We beginnen met het maken van een nieuwe custom list en geven deze de naam 'Categorieën'. We voegen hier vervolgens een extra kolom aan toe met de naam 'Parent-Title' van het type 'Lookup'. Dit Lookup-veld verwijst naar de kolom 'Title' en wordt gebruikt voor het toekennen van een bovenliggende categorie. Op deze manier kan de benodigde hiërarchie worden bereikt. Voeg alvast een aantal testwaarden toe aan de lijst. Op afbeelding 1 zie je een voorbeeld van de lijst 'Categorieën'. Maak nu een custom list aan met de naam 'Kenniscentrum'. Aan de pagina's voor het toevoegen en bewerken van items binnen deze lijst zal aan het eind van het artikel het categorieveld worden toegevoegd.

Stap 2: Custom Field Value Class

Deze klasse bevat de waarde van het Custom Field Type en roept indien noodzakelijk een zelf gedefinieerd UserControl aan. De klasse zorgt ook voor het transport van de waarde, van SharePoint, naar het UserControl en vice versa. Om dit te kunnen, moet aan SharePoint duidelijk worden gemaakt wat het dataformaat is. De klasse moet daarom enkele eigenschappen overerven van een SPField-klasse zoals SPFieldText. Deze klasse is de representatie van het veldtype Single line of text. Voor dit artikel heb ik ook gebruikgemaakt van de klasse SPFieldText. Het voordeel hiervan is dat je op eenvoudige wijze vrijwel alles (wat in tekst kan worden uitgedrukt) kunt opslaan. SPFieldText geeft dus de meeste flexibiliteit. De geselecteerde categorieën worden opgeslagen binnen een comma separated value string. Een nadeel van SPFieldText is dat er geen koppeling bestaat met een bronlijst zoals dit wel bij SPFieldLookup het geval is. Dit betekent dat er niet zomaar wijzigingen kunnen worden gemaakt in het classificatiesysteem. In ons geval is het classificatiesysteem niet tot nauwelijks aan wijzigingen onderhevig waardoor dit nadeel niet van belang is. Op het moment dat dit wel van belang zou zijn, kan worden overwogen om tevens het ID van een categorie in de comma separated value string op te slaan of om uit te zoeken hoe SPFieldLookup precies werkt. Een groot probleem (bij de base classes) is jammer genoeg de beperkte beschikbaarheid van documentatie. Naar verwachting zal dit in de toekomst verbeteren. Op het moment van schrijven worden er regelmatig nieuwe artikelen over Custom Field Types en gerelateerde onderdelen aan de MSDN Library toegevoegd. In codevoorbeeld 1 zie je hoe de Custom Field Value Class in ons geval is opgebouwd.



Afbeelding 1. Voorbeeld van de lijst met de verschillende categorieën

Stap 3: Opzetten van het Field Editor UserControl

Het Field Editor UserControl is een normaal UserControl dat getoond zal worden op de pagina's voor het toevoegen en bewerken van een item. Voor het verkrijgen van een boomstructuur wordt gebruikgemaakt van het TreeView-control. In codevoorbeeld 2 wordt het complete UserControl voor onze oplossing getoond.

```
using System;
using System.Text;
using System.Web;
using System.Collections.Generic;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;

namespace KnowledgeBrowser.CustomFieldTypes
{
  public class CategoryField : SPFieldText
  {
    public CategoryField(SPFieldCollection fields,
          string fieldName): base(fields, fieldName)
    {}

    public CategoryField(SPFieldCollection fields, string typeName,
    string displayName): base(fields, typeName, displayName)
    {}

    //Deze method kan gebruikt worden voor de extra validatie.
    public override string GetValidatedString(object value)
    {
      string myVal = value as string;
      if (myVal == null)
      {
        return String.Empty;
      }
      else
      {
        return myVal;
      }
    }
    //Deze method wordt gebruikt voor het transport van de waarde van
      het Veld.
    public override object GetFieldValue(string value)
    {
      if (String.IsNullOrEmpty(value))
      {
        return null;
      }
      return value;
    }
    //Deze method zorgt voor het tonen van het UserControl.
    public override BaseFieldControl FieldRenderingControl
    {
      get
      {
        BaseFieldControl CategoryFieldControl = new
        CategoryFieldControl();
        CategoryFieldControl.FieldName = InternalName;
        return CategoryFieldControl;
      }s
    }
  }
}
```

Codevoorbeeld 1. Custom Field Value Class

Omdat de TreeView-control gevuld moet worden met waarden uit de lijst met categorieën is er ook een 'Field Editor UserControl Class' nodig. Dit is eigenlijk een soort van codebehind voor het UserControl. Ik gebruik een comma separated value string voor het opslaan van de geselecteerde waarden binnen de TreeView. Deze comma seperated value string gebruik ik vervolgens ook weer voor het aanvinken van de juiste items bij het openen. Het vullen van het TreeView-control vind je terug binnen het volledige codevoorbeeld. op de website van het Microsoft .NET Magazine. In codevoorbeeld 3 beperk ik me tot de belangrijkste onderdelen. In codevoorbeeld 3 wordt een aantal belangrijke methods en properties getoond. Eén van de belangrijkste is Value. Via deze property communiceert het Custom Field Type met SharePoint. De waarde van het Custom Field Type wordt opgehaald en ook weer terug geschreven in het veld. De andere properties en methods spreken naar onze mening voor zich.

Stap 4: Create Custom Field Type Definition

De laatste stap is het aanmaken van een 'Custom Field Type Definition'. Dit XML-bestand zorgt ervoor dat SharePoint het Custom Field Type herkent. Er kunnen diverse eigenschappen van het Custom Field Type worden gedefinieerd. Je kunt bijvoorbeeld aangeven welk UserControl door SharePoint moet worden geladen en in welke situatie(s) deze moet worden getoond. Het is ook mogelijk binnen deze definitie invloed uit te oefenen op de presentatie van het veld met behulp van CAML (Collaborative Application Markup Language). Omdat CAML naar mijn idee buiten de scope van dit artikel valt, heb ik er voor gekozen dit niet te behandelen. In het artikel op MSDN Library over Custom Field Types wordt dit wel beknopt behandeld (zie referenties). Een Custom Field Type Definition begint altijd met de naam 'fldtypes_'. In dit geval kiezen we voor de naam 'fldtypes_category.xml'. In codevoorbeeld 4 zie je de definitie zoals gebruikt voor ons Custom Field Type.

Stap 5: Deployment

Het deployen van een Custom Field Type is, zeker als je ervaring hebt met ASP.NET-webapplicaties heel eenvoudig. Doorloop hiervoor de volgende vijf stappen.

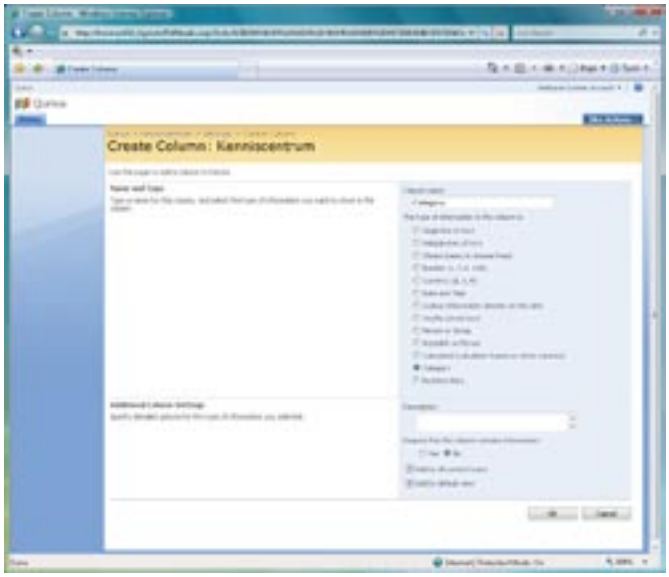
1. Compileer het project en onderteken (sign) de assembly.
2. Plaats de assembly vervolgens in de Global Assemby Cache (C:\Windows\assembly\).
3. Kopieer het UserControl (alleen de .ascx) naar \Program Files\Common Files\Microsoft Shared\web server extensions\12\template\controltemplate\.
4. Kopieer fldtypes_category.xml naar \Program Files\Common Files\Microsoft Shared\web server extensions\12\template\xml\.
5. Herstart Internet Information Services (iisreset).

```
<%@ Control Language="C#" Debug="true"%>
<%@Assembly Name="Microsoft.SharePoint, Version=12.0.0.0,
  Culture=neutral,  PublicKeyToken=71e9bce111e9429c" %>
<%@Register TagPrefix="SharePoint" Assembly="Microsoft.SharePoint,
  Version=12.0.0.0, Culture=neutral,
  PublicKeyToken=71e9bce111e9429c"
  namespace="Microsoft.SharePoint.
WebControls"%>
<SharePoint:RenderingTemplate ID="CategoryFieldControl"
  runat="server">
  <Template>
    <asp:TreeView ID="categoryTreeView" runat="server"
    ShowCheckBoxes="All" ShowLines="true" ExpandDepth="0"
    Font-Size="10">
      <Nodes>
      </Nodes>
    </asp:TreeView>
  </Template>
</SharePoint:RenderingTemplate>
```

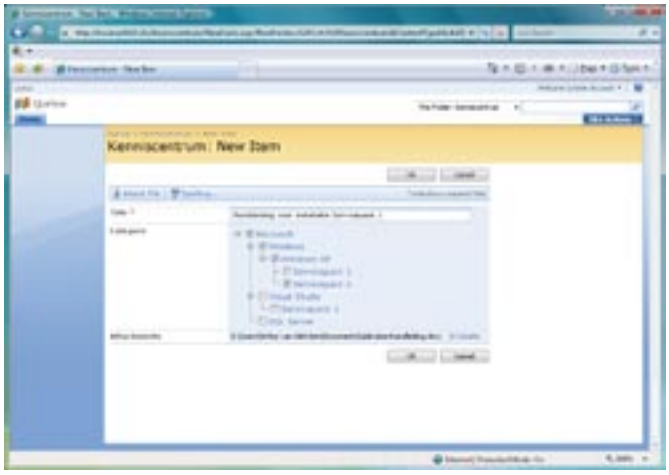
Codevoorbeeld 2. Field Editor UserControl

```
amespace KnowledgeBrowser.CustomFieldTypes
{
  public class CategoryFieldControl : BaseFieldControl
  {
    TreeView categoryTreeView = new TreeView();
    TreeNode node = new TreeNode();
    DataTable catagoryDataTable = new DataTable();

    protected override string DefaultTemplateName
    {
      get
      {
        return "CategoryFieldControl";
      }
    }

    public override object Value
    {
      get
      {
        // Teruggeven van een comma separated value string met
        geselecteerde waardes uit de TreeView.
        return retValue;
      }
      set
      {
        selecteditems = this.ItemFieldValue.ToString().Split(';');
        foreach (string selecteditem in selecteditems)
        {
          //Aanvinken van geselecteerde treenodes
        }
      }
    }

    //Binnen deze method kunnen voor verschillende situaties,
    verschillende controls/opties worden getoond.
    protected override void CreateChildControls()
    {
      if (Field == null) return;

      if (ControlMode == SPControlMode.Display)
      {
        return;
      }

      if (!Page.IsPostBack)
      {
        try
        {
          base.CreateChildControls();
          categoryTreeView = (TreeView)TemplateContainer.FindControl
          ("categoryTreeView");
          if (categoryTreeView != null)
          {
            LoadTreeview(); //Vullen van de TreeView
          }
          else
          {
            throw new ArgumentException("De categoryTreeView kan niet
            worden gevonden. Het aanmaken van treenodes is niet gelukt.");
          }
          return;
        }
        catch (Exception ex)
        {
          throw new ArgumentException(ex.Message);
        }
      }
      else
      {
        base.CreateChildControls();
        categoryTreeView = (TreeView)TemplateContainer.FindControl
        ("categoryTreeView");
        if (categoryTreeView == null)
        {
          throw new ArgumentException("De categoryTreeView kan niet
          worden gevonden. Mogelijk is het .ascx bestand corrupt.");
        }
      }
    }
  }
}
```

Codevoorbeeld 3. Field Editor UserControl Class



Afbeelding 2. Aanmaken van een nieuwe kolom binnen de document library.



Afbeelding 3. Custom Field Type in actie.

Op het moment dat je nu een nieuwe kolom aan een lijst toevoegt (de eerder gemaakte lijst voor documenten en informatie), zie je dat er naast de gebruikelijke fieldtypes ook het nieuwe type 'Category' kan worden gekozen; zie afbeelding 2.

We voegen de kolom 'Categorie' toe aan de lijst. We kunnen de functionaliteit van het geheel testen door een nieuw item aan de lijst toe te voegen. In afbeelding 3 zie je het Custom Field Type eindelijk in actie.

Tot slot

Met Custom Field Types is natuurlijk veel meer mogelijk dan dat ik binnen het beperkte aantal pagina´s van dit artikel kan laten zien. Door bijvoorbeeld het zojuist gemaakte Custom Field Type ook binnen andere lijsten te gebruiken, ontstaat er een duidelijke en logische structuur op de website wat vooral het zoeken naar verschillende items erg eenvoudig maakt. Het gebruik van Custom Field Types kan een oplossing bieden voor in ieder geval een drietal beperkingen van SharePoint. Het zelf invloed kunnen uitoefenen op de presentatie van een veld, het toepassen van een eigen datastructuur of het toevoegen van extra validatiemogelijkheden.
We hopen dat de gedachte achter dit artikel je aanzet tot het experimenteren met Custom Field Types. Als je dit doet, dan zul je zeker enkele nieuwe mogelijkheden ontdekken waarmee je beperkingen in jouw specifieke situatie met succes kunt oplossen.

```
<?xml version="1.0" encoding="utf-8" ?>
<FieldTypes>
  <FieldType>
    <Field Name="TypeName">Category</Field>
    <Field Name="ParentType">Text</Field>
    <Field Name="TypeDisplayName">Category</Field>
    <Field Name="TypeShortDescription">Category</Field>
    <Field Name="ShowInDisplayForm">TRUE</Field>
    <Field Name="ShowInEditForm">TRUE</Field>
    <Field Name="ShowInListSettings">TRUE</Field>
    <Field Name="ShowInNewForm">TRUE</Field>
    <Field Name="ShowInVersionHistory">TRUE</Field>
    <Field Name="ShowInViewForms">TRUE</Field>
    <Field Name="FieldTypeClass">KnowledgeBrowser.CustomFieldTypes.
CategoryField,Qurius.KnowledgeBrowser.CustomFieldTypes,Version=1.0.0.0,
Culture=neutral,PublicKeyToken=86616b56abf8c5dd</Field>
  </FieldType>
</FieldTypes>
```

Codevoorbeeld 4. Custom Field Type Definition

Arthur van Adrichem is als technisch consultant werkzaam bij Qurius AS (www.qurius.nl) in Rijswijk. Zijn interesses liggen vooral op het gebied van Microsoft Dynamics CRM en Microsoft Office SharePoint Server 2007. Voor vragen en opmerkingen is hij te bereiken via arthur.van.adrichem@qurius.nl.

Referenties

MSDN over Custom Field Types: http://msdn2.microsoft.com/en-us/library/ms446361.aspx
MSDN over SPField classes: http://msdn2.microsoft.com/en-us/library/microsoft.sharepoint.spfield.aspx
Complete code: http://www.softwarestraat.net/magazine/september/CustomFieldTypes.aspx