Ton Stegeman is technisch SharePoint-consultant voor e-office

Een policy op maat

SCHRIJF JE EIGEN SHAREPOINT INFORMATION MANAGEMENT POLICY

Microsoft Office SharePoint Server 2007 biedt uitgebreide mogelijkheden informatie in een organisatie beter te managen, door onder andere het gebruik van policies (beleidsregels). In dit artikel worden – aan de hand van een voorbeeld – de stappen behandeld voor het ontwikkelen van een policy op maat.

Met de komst van Microsoft Office SharePoint Server 2007 (MOSS) is het SharePoint Technologies-platform uitgebreid met het information management policy-framework. De policies in dit framework ondersteunen het informatiemanagementbeleid in een organisatie. Bij een SharePoint-installatie worden vier policies geleverd:

- **Expiration –** deze policy ondersteunt het records-managementbeleid. Aan het eind van de levenscyclus wordt de content op een gecontroleerde manier gearchiveerd of verwijderd.
- **Auditing –** een policy die records-management en 'compliancy' kan ondersteunen. Met deze policy is het mogelijk een controlesysteem voor content op te zetten. Alle toegang tot de content (lezen en schrijven) kan worden gelogd.
- **Documentlabels –** Een policy die er voor zorgt dat belangrijke informatie die wordt verzameld in de metadata van documenten ook in het document zelf wordt opgenomen.
- **Documentbarcodes –** voor ieder document waarvoor deze policy geldt, wordt automatisch een barcode gegenereerd. Het document is ook weer terug te vinden met deze barcode.

Deze policies worden voornamelijk gekoppeld aan een SharePoint-contenttype. Het policy-framework zorgt er voor dat elk document (of elk ander SharePoint-item) automatisch aan de regels van de policy voldoet. Het is ook mogelijk de policies direct aan een document-library of een lijst te koppelen. Bovendien kunnen ze op het hoogste niveau in een site collection, de top level-site gedefinieerd worden. Dit maakt de policies algemeen beschikbaar binnen alle sites. Het policy-framework in Microsoft Office SharePoint Server kan verder worden uitgebreid. Dat betekent dat het mogelijk is policies op maat te maken. Hierdoor sluit SharePoint naadloos aan op het informatiebeleid in jouw organisatie. Een voorbeeld uit de praktijk is een policy voor contractmanagement. Deze policy verstuurt reminders naar beheerders van contracten als de einddatum van een contract nadert. De SharePoint Server 2007 SDK bevat uitgebreidere informatie. De ECM Starter Kit bevat een uitgewerkt voorbeeld van een policy.

Scenario

In dit artikel worden alle stappen van het maken van een eigen policy doorlopen. Dit wordt gedaan aan de hand van een voorbeeld. Hierin wordt de 'policy of truth' geïmplementeerd. Deze policy helpt met het monitoren van documenten waarin gebruikers 'de waarheid' over bepaalde onderwerpen schrijven. Tijdens het configureren van deze policy kan een beheerder een aantal keywords opgeven. De policy kijkt of een gebruiker een van deze keywords in de metadata van een document zet, gecombineerd met het woord 'truth'. Als dat het geval is, stuurt de policy het document naar een records-center. Vanuit dit records-center wordt centraal beheerd welke documenten er in de organisatie geproduceerd worden die als 'de waarheid' beschouwd kunnen worden. Het document zelf krijgt een extra eigenschap waar de gebruikers aan kunnen zien dat het document centraal wordt gemanaged. Dit voorbeeld is niet gebaseerd op een scenario uit de dagelijkse praktijk, maar toont wel alle aspecten van het ontwikkelen van een policy. De codevoorbeelden in dit artikel zijn slechts snippets die een beperkt stuk actie uitvoeren. Error handling en logging zijn niet in deze code verwerkt.

Stap 1 - De policy-feature

De basis van een SharePoint-policy is de PolicyFeature. Deze maken we door de PolicyFeature-interface te implementeren. We implementeren deze interface in een nieuwe class genaamd PolicyOfTruth. De assembly deployen we naar het GAC en moet daarom een strong name hebben. De interface is te vinden in de namespace "Microsoft.Office.RecordsManagement.InformationPolicy" in de Microsoft.Office.Policy-assembly. In de interface die we implementeren, zijn Register en UnRegister de belangrijkste methods. Register wordt door het policy-framework aangeroepen op het moment dat een policy aan een contenttype wordt gekoppeld. Hierin koppelen we eerst de event-handler(s) aan het contenttype. In ons scenario koppelen we slechts een handler voor het ItemUpdated-event aan het contenttype. In codevoorbeeld 1 is 'ct' een object van het type SPContentType dat als parameter van de Register-method wordt meegegeven.

Daarna voegen we extra benodigde velden aan het contenttype toe. Er wordt een nieuwe site-column gemaakt en deze wordt toegevoegd aan het contentype. Zie hiervoor codevoorbeeld 2.
In dit extra veld wordt de datum opgeslagen bij een document als dat document in de centrale repository wordt opgeslagen. Deze waarde kan door een gebruiker niet worden aangepast.
De methode 'UnRegister' van de policy-feature ontkoppelt de event-handlers van het contenttype en verwijdert eventueel het extra veld. Twee andere interessante methodes op de policy-feature zijn 'ProcessListItem' en 'ProcessListItemOnRemove'. De eerste wordt aangeroepen voor listitems die al in SharePoint aanwezig waren, voordat de policy werd gekoppeld. Dit maakt het mogelijk ook bestaande content 'compliant' met de policy te krijgen.

```
Assembly assembly = Assembly.GetExecutingAssembly();
SPEventReceiverDefinition eventReceiver = ct.EventReceivers.Add();
eventReceiver.Name = "Policy of Truth";
eventReceiver.Type = SPEventReceiverType.ItemUpdated;
eventReceiver.SequenceNumber = 200;
eventReceiver.Assembly = assembly.FullName;
eventReceiver.Class = "TST.POC.PolicyFeatures.PolicyOfTruthHandler";
eventReceiver.Update();
```

Codevoorbeeld 1.

Stap 2 – Configuratie

Een SharePoint-policy heeft een property 'CustomData' waarin specifieke configuratiegegevens voor de policy kunnen worden opgeslagen. Deze worden opgeslagen in een XML-string. In ons scenario slaan we daarin de keywords op. Deze keywords bepalen of een document voldoet aan de voorwaarde om te worden toegevoegd aan de centrale repository. Deze eigenschappen worden door een beheerder ingevuld tijdens de configuratie van de policy. Voor onze 'Policy of Truth' ziet dat eruit als in afbeelding 1. Onder de algemene policy-gegevens zien we de specifieke instellingen voor onze policy.

Afbeelding 1. Configuratiescherm van een policy

We maken dit mogelijk door aan onze class-library een object 'PolicyOfTruthSettings' toe te voegen. Dit object overerft van 'CustomSettingsControl', uit de eerder genoemde namespace. Daarnaast maken we een ASCX-control die de user-interface van onze policy-settings-control rendert. In ons geval rendert de ASCX een label en een textbox, genaamd 'TextBoxKeywords'. In de header van de ASCX verwijzen we het attribuut 'Inherits' van het element 'Control' naar 'TST.POC.PolicyFeatures.PolicyOfTruthSettings'. Deze ASCX deployen we naar de "..\12\TEMPLATE\LAYOUTS"-folder van SharePoint. In de 'CustomSettingsControl' implementeren we een aantal abstracte properties en methods. De belangrijkste zijn:

- **Property CustomData.** In de getter van deze property genereren we de XML-string met de configuratiewaarde(n), die we uit de controls lezen. Zie hiervoor codevoorbeeld 3.
- **LoadPostData.** Hierin bewaren we de CustomData in de editor-control, zodat deze met de overige policy-gegevens worden opgeslagen. Codevoorbeeld 4 toont de code daarvoor.
- **OnLoad.** In de OnLoad, die is weergegeven in codevoorbeeld 5, wordt de XML ingelezen en worden de waarden van de controls in onze ASCX-controls gezet.

Door op de policy-feature de methodes 'OnCustomDataChange' en 'OnGlobalCustomDataChange' te implementeren, kunnen we actie ondernemen op de documenten als de configuratie van onze policy gewijzigd wordt door een beheerder. In ons voorbeeld betekent dit dat de keywords gewijzigd zijn en moeten we alle documenten opnieuw checken om de centrale repository up-to-date te houden. In dit voorbeeld zijn ze niet geïmplementeerd.

Stap 3 - Registreren van de policy

De volgende stap is het registreren van de policy. Dit doen we vanuit code. De code uit codevoorbeeld 7 wordt geïmplementeerd in een SharePoint-feature. We maken een object SPFeatureReceiver, dat we vervolgens kunnen installeren en activeren. Eerst schrijven we een file manifest.xml met daarin de benodigde gegevens voor de registratie. De voorbeeld-XML voor onze policy is weergegeven in codevoorbeeld 6.

```
string fieldName = "SentToTruthRepository";
foreach (SPFieldLink link in contentType.FieldLinks)
  if (link.Name == fieldName)
    return;
SPField repositoryField = null;

using (SPWeb web = contentType.ParentWeb)
{
  repositoryField = web.AvailableFields.GetFieldByInternalName
(fieldName);
  if (repositoryField == null)
  {
    string xml = "<Field Name=\"SentToTruthRepository\" ";
    xml += "FromBaseType=\"FALSE\" Type=\"DateTime\" ";
    xml += "DisplayName=\"Sent to truth repository\" ";
    xml += "Required=\"TRUE\" Format=\"DateTime\" ";
    xml += "ReadOnly=\"TRUE\" Group=\"Policy Columns\" />";
    string newField = web.Fields.AddFieldAsXml(xml);
    repositoryField = web.Fields.GetFieldByInternalName(newField);
  }
}
```

Codevoorbeeld 2.

In het element ConfigPage zetten we de naam van de ASCX-control uit de vorige stap. De AssemblyName- en ClassName-elementen bevatten de verwijzing naar de policy-feature. De XML-file laden we in onze feature-receiver, waarin we de policy aan de catalog toevoegen. Zie codevoorbeeld 7.

Eerst kijken we hierin of de policy al is geïnstalleerd. Als dat nog niet het geval is, wordt de XML uit het manifest gevalideerd met 'ValidateManifest'. Vervolgens wordt de policy toegevoegd aan de PolicyFeatureCollection. Deze PolicyFeatureCollection is beschikbaar in de namespace 'Microsoft.Office.RecordsManagement.InformationPolicy'. Het registreren van onze policy in deze collectie maakt de policy beschikbaar in iedere webapplicatie die op de SharePoint-farm aanwezig is.
Het verwijderen van de policy van het systeem kan heel simpel met de code in codevoorbeeld 8. De beste plek om deze code te implementeren, is in dezelfde feature, bijvoorbeeld in de method FeatureUninstalling.

```
public override string CustomData
{
  get
  {
    XmlDocument doc = new XmlDocument();
    XmlElement rootNode = doc.CreateElement("data");
    doc.AppendChild(rootNode);
    XmlElement keywordsNode = doc.CreateElement("keywords");
    rootNode.AppendChild(keywordsNode);
    keywordsNode.InnerText = TextBoxKeywords.Text;
    _customData = doc.InnerXml;
    return _customData;
  }
  set {_customData = value;}
}
```

Codevoorbeeld 3.

```
public override bool LoadPostData(string postDataKey,
  System.Collections.Specialized.NameValueCollection values)
{
  string oldData = this.CustomData;
  string newData = values[postDataKey];
  if (oldData!=newData)
  {
    this.CustomData = newData;
    return true;
    }
  return false;
}
```

Codevoorbeeld 4.

```
protected override void OnLoad(EventArgs e)
{
  base.OnLoad(e);
  if ((base.IsPostBack) || (string.IsNullOrEmpty(_customData)))
  {
    return;
  }
  using (XmlReader reader = XmlReader.Create(
    new System.IO.StringReader(_customData)))
  {
    reader.ReadStartElement("data");
    reader.ReadStartElement("keywords");
    TextBoxKeywords.Text = reader.ReadString();
    reader.ReadEndElement();
    reader.ReadEndElement();
  }
}
```

Codevoorbeeld 5.

Stap 4 –De records-repository-handler

De volgende stap is het implementeren van een class (genaamd RepositoryHandler) in de class-library die ervoor zorgt dat het document wordt verstuurd naar het records-center als het voldoet aan de vastgestelde voorwaarden. Deze handler gebruiken we vanuit de event-handler die we hebben geregistreerd in de eerste stap. Eerst voegen we een webreference aan ons project toe. De url hiervan zetten we naar de de url van de webservice van het SharePoint Records Center: http://[server]/_vti_bin/OfficialFile.asmx
Hierin is [server] de url van een SharePoint-site die gebaseerd is op de 'Records Center' site-template. In de aanroep van deze repository-handler geven we het SPListItem-object uit het ItemUpdated-event mee. Als eerste moeten we de keywords uit de policy opvragen, zoals die door de beheerder zijn ingesteld.

In codevoorbeeld 9 vragen we eerst alle policies op die gekoppeld zijn aan het contenttype. Vervolgens vragen we daaruit de policy op met het id van onze Policy of Truth. Dit is het id waarmee de policy-feature is geregistreerd in de vorige stap. Door de waarde van de CustomData uit te lezen, vragen we de keywords op. De repository-handler controleert vervolgens of een van de keywords voorkomt in de metadata van het listitem. Als dat het geval is, wordt het item door de code in codevoorbeeld 10 naar het records-center gestuurd.
De repository-handler geeft aan de event-handler terug dat het document succesvol is toegevoegd aan de centrale repository. Voor dit voorbeeld is er geen speciale record-routing opgezet in het records-center. Het document wordt aan de repository toegevoegd aan de 'Unclassified Records'.

```
<?xml version="1.0" encoding="utf-8" ?>
<p:PolicyFeature id="TST.POC.PolicyFeatures.PolicyOfTruth"
  xmlns:p="urn:schemas-microsoft-com:office:server:policy"
  group="Policy">
  <p:LocalizationResources>dlccore</p:LocalizationResources>
  <p:Name>Policy of Truth</p:Name>
  <p:Description>This policy manages our 'truth'</p:Description>
  <p:Publisher>Ton Stegeman</p:Publisher>
  <p:ConfigPage>policyoftruthsettings.ascx</p:ConfigPage>
  <p:ConfigPageInstructions>You can add your keywords here.
  </p:ConfigPageInstructions>
  <p:AssemblyName>TST.POC.PolicyOfTruth, Version=1.0.0.0,
  Culture=neutral, PublicKeyToken=503edd7b21a430b3</p:AssemblyName>
<p:ClassName>TST.POC.PolicyFeatures.PolicyOfTruth</p:ClassName>
</p:PolicyFeature>
```

Codevoorbeeld 6.

```
PolicyFeatureCollection policyFeatures = PolicyCatalog.FeatureList;
foreach (PolicyFeature policyFeature in policyFeatures)
{
  if (policyFeature.Id=="TST.POC.PolicyFeatures.PolicyOfTruth")
  {
    return;
  }
}
string manifest = System.IO.File.ReadAllText("manifest.xml");
PolicyFeature.ValidateManifest(manifest);
PolicyFeatureCollection.Add(manifest);
```

Codevoorbeeld 7.

Stap 5 - Implementeer de event-handler

De laatste stap is het maken van een normale SharePoint-list-event-handler. In codevoorbeeld 11 maken we een event-handler die gebruikmaakt van de handler die we in stap 4 hebben gemaakt. We maken een class die overerft van SPItemEventReceiver en overerven de method; zie codevoorbeeld 11.
Voor deze democode gebruiken we alleen het event ItemUpdated. Als de repository-handler het document heeft verstuurd naar het records-center, wordt de huidige datum en tijd opgeslagen in het read-only-veld 'Sent to truth repository' bij het oorspronkelijke document. Om te voorkomen dat deze laatste update-actie wederom ons event triggert, starten we de event-handler met het aanroepen van 'DisableEventFiring'.

Natuurlijk kan het ook anders

In dit artikel zijn alle stappen in het ontwikkelproces van een information management policy de revue gepasseerd. De basis van een information management policy is de policy-feature. In ons voorbeeld is dat niet veel meer dan een registratie van een event-handler op een contenttype. Dit kan dus net zo goed met alleen een event-handler worden opgelost. Een andere mogelijkheid is het bouwen van een workflow. Hieronder nog wat overwegingen waarom je wel, of juist niet, voor een policy zou kiezen.

- **Flexibiliteit**. Een policy is flexibeler dan een event-handler. Sitebeheerders kunnen zelf de policies gebruiken op de plaatsen waar dat nodig is. Dat kan zonder tussenkomst van een technisch beheerder.
- **Interactie**. Als er tijdens het proces interactie met de gebruiker moet plaatsvinden, is een workflow een beter alternatief. Let er wel op dat het bouwen en registreren van zulke workflows geen eenvoudige klus is.

```
PolicyFeatureCollection policyFeatures = PolicyCatalog.FeatureList;
foreach (PolicyFeature policyFeature in policyFeatures)
{
  if (policyFeature.Id == "TST.POC.PolicyFeatures.PolicyOfTruth")
  {
  PolicyFeatureCollection.Delete(policyFeature.Id);
  return;
  }
}
```

Codevoorbeeld 8.

```
Policy policy = Policy.GetPolicy(item.ContentType);
PolicyItem policyItem = policy.Items[PolicyOfTruth.PolicyId];
string keywords = string.Empty;
using (XmlReader reader = XmlReader.Create(
  new System.IO.StringReader(policyItem.CustomData)))
{
  reader.ReadStartElement("data");
  reader.ReadStartElement("keywords");
  keywords = reader.ReadString();
  reader.ReadEndElement();
  reader.ReadEndElement();
}
```

Codevoorbeeld 9.

```
TST.POC.PolicyFeatures.Repository.RecordsRepository repository =
  new Repository.RecordsRepository();
repository.Credentials = System.Net.CredentialCache.DefaultCredentials;
repository.PreAuthenticate = true;

Repository.RecordsRepositoryProperty[] repositoryProperties =
  new Repository.RecordsRepositoryProperty[0];

byte[] doc = item.File.OpenBinary();
string result = repository.SubmitFile(doc, repositoryProperties,
  "Unclassified Records", item.Url, item.Web.CurrentUser.Name);

result = string.Format("<Result>{0}</Result>", result);
XmlDocument xml = new XmlDocument();
xml.LoadXml(result);
XmlElement root = xml.DocumentElement;
string resultCode = root.SelectSingleNode("ResultCode").FirstChild.
Value;
return (resultCode=="Success");
```

Codevoorbeeld 10.

```
public override void ItemUpdated(SPItemEventProperties properties)
{
  DisableEventFiring();
  RepositoryHandler repository = new RepositoryHandler();
  if (repository.HandleListItem(properties.ListItem))
  {
    string truthFieldName = "Sent to truth repository";
    properties.ListItem[truthFieldName] = DateTime.Now;
    properties.ListItem.Update();
  }
  EnableEventFiring();
}
```

Codevoorbeeld 11.

- **Configuratie.** Het policy-framework biedt de mogelijkheid om tijdens de configuratie input van de gebruiker te vragen. Dit kan met een eenvoudige ASCX-control. Voor een workflow is dit een stuk lastiger en voor een event onmogelijk.
- **Uitzonderingen.** Het policy-framework biedt de gebruiker de mogelijkheid in noodgevallen items buiten de policy om te behandelen. Dit heet 'Exempt from policy'. Met alleen een event is dit onmogelijk.

Ton Stegeman is MVP Office SharePoint Server en werkt als technisch SharePoint-consultant voor e-office (www.e-office.com). Zijn e-mailadres is ton.stegeman@e-office.com. Op zijn weblog is nog veel meer technische SharePoint-informatie te vinden: http://www.sharepointblogs.com/tonstegeman.

Referenties

Het policy-framework: http://msdn2.microsoft.com/en-us/library/ms499244.aspx
MOSS SDK en ECM Starter Kit: http://www.microsoft.com/downloads/details.aspx?familyid=6D94E307-67D9-41AC-B2D6-0074D6286FA9&displaylang=en
Microsoft Records Management Team Blog: http://blogs.msdn.com/recman/default.aspx
De code uit dit artikel: http://www.sharepointblogs.com/tonstegeman/attachment/19518.ashx