Astrid Hackenberg
Astrid is werkzaam als trainer/coach bij Class-A. Zij heeft als ontwikkelaar, trainer en consultant gewerkt bij een Microsoft Partner/CTEC en daarna bij Microsoft zelf. Veel ontwikkelteams zijn door haar getraind en gecoacht. Astrid's specialisatie is data: data-acces, data-uitwisseling, data-architectuur en datatechnologieën. Sinds 2001 werkt zij met .NET
astrid.hackenberg@class-a.nl
www.class-a.nl

Globalisatie in ASP.NET

HOE KAN JE EEN ASP.NET-APPLICATIE GESCHIKT MAKEN VOOR INTERNATIONAAL GEBRUIK?

Dit artikel behandelt het vraagstuk: "Hoe kan ik mijn ASP.NET-applicatie gebruiken in verschillende landen en met verschillende talen, zonder dat ik de applicatie meermalen bouw?" Dit artikel gaat in op wat .NET ons biedt aan globalisatieservices en hoe die gebruikt kunnen worden bij het bouwen van webapplicaties.

Wat is het probleem? Het marktgebied van veel bedrijven gaat in toenemende mate over grenzen heen. Er is de afgelopen jaren een afzetkanaal bijgekomen waardoor er naast multinationals ook vele bedrijven zijn gekomen die buiten de eigen regio opereren; Internet. (Web)applicaties en websites moeten zich dan ook steeds vaker kunnen aanpassen aan een land, de klant of bedrijfslocatie. Kortom, onze applicaties moeten geschikt zijn voor internationaal gebruik. Hiervoor moet de applicatie cultuurneutraal en taalneutraal worden gebouwd. Dan gaan we lokaliseren. Eén aspect daarvan is meertaligheid, maar er is meer. Ook getal- en datumweergave, formattering, iconen, plaatjes, enzovoort kunnen verschillend zijn. Lokaliseren houdt in dat je rekening houdt met dialecten en culturele verschillen, zelfs als men dezelfde taal spreekt. We moeten lokaliseren om een applicatie te bouwen die internationaal bruikbaar is.

Stel je voor dat je op vakantie gaat naar Portugal en een auto wilt huren. Het verhuurbedrijf heeft een website waarmee je een auto kunt reserveren; alleen jij kent geen Portugees. Dit geldt voor de meeste toeristen. Het verhuurbedrijf geeft daarom de mogelijkheid om op de website een taal naar keuze te selecteren. In veel gevallen zijn hiervoor de webpagina's gekopieerd en zijn de teksten op de pagina vertaald. Dit is uiteraard veel werk, maar maakt onderhoud bovendien tijdrovend en duur. Daarnaast wil je bijvoorbeeld ook een datum kunnen invoeren in een voor jou logisch formaat. Met vertalen alleen zijn ze er dus niet, ze moeten de hele applicatie op jouw cultuur afstemmen.

Wat biedt de .NET infrastructuur?

.NET biedt ons een aantal services om onze applicaties te lokaliseren. De twee belangrijkste aspecten en onderdelen daarbij zijn resources en cultures. Deze services vind je terug in de namespaces System.Globalization en System.Resources. De System.Globalization-namespace bevat classes die cultuurgerelateerde informatie definiëren. De System.Resources-namespace bevat classes waarmee cultuurspecifieke resources kunnen worden gemaakt, gelezen en opgeslagen.

Cultures

Culture verwijst in het .NET Framework naar de taal van de gebruiker in combinatie met zijn locatie. Door een cultuur te specificeren bepaal je de voorkeuren voor het formatteren van gegevens zoals tekst, data en getallen. Een culture wordt bepaald door twee ISO-codes. Ten eerste de 2-letterige code voor namen van talen en ten tweede de 2-letterige code voor namen van landen. Een voorbeeld: de culture voor Nederlands in Nederland is "nl-NL", de culture voor Nederlands in België is "nl-BE". Bij het specificeren van een cultuur is het land optioneel. Dus de culture voor Nederlands is "nl" ongeacht het land, dit is de neutral culture. Er bestaat ook een invariant culture, wat wil zeggen dat geen specifieke culture is gedefinieerd. In de framework-documentatie is de volledige lijst van bruikbare cultures opgenomen. Als je een CultureInfo-object creëert voor een specifieke cultuur dan bevat deze ook al voorgedefinieerde FormatInfo-objecten. Bijvoorbeeld een DateTimeFormatInfo-object die onder andere bepaalt hoe een DateTime-instantie de datum weergeeft. Uiteraard kun je deze aanpassen of je eigen FormatInfo-classes bouwen.

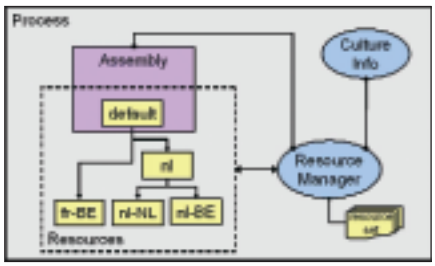
Resources

Een resource is data die bij een applicatie hoort, maar geen executable code is. Resources worden samen met de appli-

Resources kunnen in verschillende formaten worden opgeslagen.

Formaat	Extensie	Gebruik
Tekst	.txt	Kan worden beheert met iedere teksteditor Werkt met Stream/TextReader of Stream/TextWriter Kan alleen tekst bevatten
Xml	.resx	Kan worden beheerd met iedere tekst- of XML-editor Werkt met ResxResourceReader en ResxResourceWriter Kan objecten en/of tekst bevatten
Binair	.resources	Werkt met ResourceReader, ResourceWriter en ResourceManager Kan objecten en/of tekst bevatten
Gecompileerd	resources.dll	Als assembly Werkt met ResourceManager Kan objecten en/of tekst bevatten

catie gedeployed. Door deze data in een resourcefile op te slaan kan deze worden aangepast en gedeployed, zonder dat hiervoor de applicatie (opnieuw) moet worden gecompileerd.

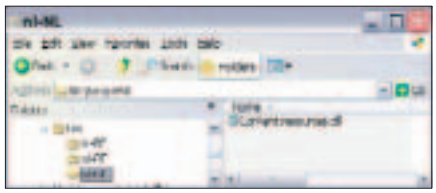


Afbeelding 1. Globalisatiearchitectuur

De resourcemanager geeft je op een eenvoudige wijze toegang tot de cultuurcorrecte resources. De resourcemanager is gekoppeld aan een assembly en cultuure, en zoekt op basis daarvan de juiste resourceset in de beschikbare resourcefiles. Hierbij wordt steeds gezocht naar de meest specifieke resourceset die beschikbaar is volgens een fallback-scenario. Eerst wordt gezocht naar een specifieke resourceset voor taal- en landcombinatie, dan naar een neutral resourceset voor alleen het land en uiteindelijk naar de default- of invariant resourceset. Deze architectuur is schematisch weergegeven in afbeelding 1.
Tekst en XML-resourcefiles zijn je sources die je gebruikt voor vertaling en lokalisatie.

Deployment in .NET

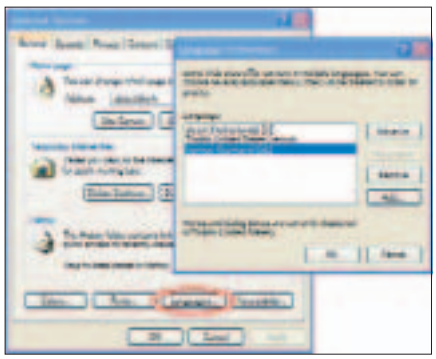
Resources kunnen worden gedeployed als binaire files of in assemblies. Om runtime gebruik te maken van binaire files moeten naam en directory expliciet opgegeven worden. Het voordeel is dat dit je veel flexibiliteit oplevert, het nadeel is dat het fallback-scenario niet werkt en de files worden gelockt. Het ASP.NET 'live update' mechanisme werkt dan dus niet meer. Om gebruik te maken van het fallback-scenario van de resourcemanager moeten tot assembly gecompileerde resources worden gebruikt. De resources voor de default-culture moeten in de main assembly worden opgenomen en de resources voor alle andere cultures in satellite-assemblies. Bij het opzoeken van de juiste resourceset tijdens het fallback-proces wordt allereerst gezocht in de Global Assembly Cache. Dit gaat op basis van de strong name, waarvan de culture onderdeel uitmaakt. Daarna wordt in de directory-tree van de executing assembly gezocht naar een directory met de juiste culture. In deze directory wordt dan gezocht naar een satellite-assembly met de juiste resourceset. In afbeelding 2 is een directory-structuur weergegeven waarin de cultures "nl-NL", "nl-BE" en "fr-BE" zijn opgenomen.



Afbeelding 2. Directory-structuur satellite-assemblies

Hoe kunnen lokalisatieservices worden toegepast in ASP.NET?

Standaard zal .NET er vanuit gaan dat we de culture willen gebruiken volgens de regionale instellingen in Windows. Dit kan prima werken in een WinForms-applicatie als we de gebruikersinstellingen simpelweg willen overnemen. Bij een webapplicatie hebben we echter vaak te maken met onbekende clients, we weten niet veel van de omgeving van de gebruiker. Wel kan je een aantal browser-instellingen opvragen zoals de browsertaal van de gebruiker (bij gebruik van de Internet Explorer). Zie afbeelding 3.



Afbeelding 3. Instellen van taal in Internet Explorer

Met de HttpBrowserCapabilities-class uit de system.web namespace is het mogelijk de browsertaal op te vragen via HTTP_ACCEPT_LANGUAGE. Het resultaat van voorbeeldcode 1 voor de browserinstellingen in afbeelding 3 is "nl; en-us; q=0.7,de;q=0.3".
Dit werkt goed in een intranetomgeving, maar helaas niet voor 100% op het internet. Veel gebruikers hebben de instelling niet juist staan of in het geval van een internetcafé bijvoorbeeld staat de instelling waarschijnlijk op Engels.
De gebruiker zal ons dus moeten vertellen

```
<%@ Page language="c#"%>
<%@ Import Namespace="System.Web" %>

<HTML>
   <HEAD></HEAD>
   <Body>
      <% HttpBrowserCapabilities browserObj;
      browserObj = Request.Browser;%>

      Toon de Browser Language:
      <%=Request.ServerVariables["http_accept_language"]%>
   </body>
</HTML>
```

Voorbeeldcode 1. Opvragen van de browsertaal in ASP.NET

welke culture hij/zij wil, anders worden de regionale instellingen van de webserver gebruikt. Een mogelijkheid is om de gebruiker, zoals eerder beschreven voor de autoverhuursite, de taal en/of het land te laten selecteren, maar we kunnen natuurlijk ook een profiel van de gebruiker opslaan. Aangezien de webpagina's op de server door ASP.NET worden gegenereerd, moet je de resourcefiles ook op de webserver deployen.

Het implementeren van lokalisatie is geen onderdeel van het ASP.NET Framework. Je zult alles zelf moeten coderen en de Web-Form Designer heeft geen multiple culture-support. Dit in tegenstelling tot een Windows-applicatie. De WinForms Designer ondersteunt wel het gebruik van verschillende cultures. Er worden automatisch resourcefiles gecreëerd en er wordt resourcemanager-code gegenereerd.

Hoe gebruik je een resourcemanager?

In voorbeeldcode 2 zie je hoe je een resourcemanager kunt creëren en gebruiken. Op de resourcemanager is een GetString() en GetObject() methode beschikbaar die je kunt aanroepen om te lokaliseren. Dit is enorm veel werk. Je moet voor iedere waarde die je wilt lezen een statement coderen. Bovendien is deze code constant aan wijziging onderhevig. Als je een control toevoegt of verwijdert, of een extra eigenschap wilt lokaliseren, moet je de code aanpassen. In de volgende paragraaf wordt een methode beschreven om dit te automatiseren.

```
Private Sub Page_PreRender(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles MyBase.PreRender

  Dim _resourceManager As Resources.ResourceManager
  Dim _type As Type

  'Retrieve the type information of the ASPX page codebehind class
  _type = Page.GetType().BaseType

  'Create a resource manager for the ASPX page
  _resourceManager = New Resources.ResourceManager(_type.FullName, _
  _type.Assembly)

  'Get the lokalized values for the current culture

    'Translate the simple controls
    Me.titelLabel.Text = _resourceManager.GetString("titelLabel.Text")
    Me.selectButton.Text = _resourceManager.GetString( _
    "selectButton.Text")
    Me.availableCheck.Text = _resourceManager.GetString( _
    "availableCheck.Text")
    Me.allCarsCheck.Text = _resourceManager.GetString( _
    "allCarsCheck.Text")

    'Translate the datagrid headers
    Me.carsList.Columns(1).HeaderText = _resourceManager.GetString( _
    "carsList.Column1.Header")
    Me.carsList.Columns(2).HeaderText = _resourceManager.GetString( _
    "carsList.Column2.Header")
    Me.carsList.Columns(3).HeaderText = _resourceManager.GetString( _
    "carsList.Column3.Header")
    Me.carsList.Columns(4).HeaderText = _resourceManager.GetString( _
    "carsList.Column4.Header")

    'Translate the datagrid buttons
    Dim _gridItem As DataGridItem
    For Each _gridItem In Me.carsList.Items
        CType(_gridItem.Cells(1).Controls(0), LinkButton).Text = _
        _resourceManager.GetString("carsList.ButtonShowDetails.Text")
        CType(_gridItem.Cells(6).Controls(0), LinkButton).Text = _
        _resourceManager.GetString("carsList.ButtonEdit.Text")
    Next

End Sub
```

Voorbeeldcode 2. Gebruik resourcemanager

Bij het gebruik van een resourcemanager in een ASPX-pagina is er een aantal bijzonderheden waarmee je rekening moet houden. Allereerst het moment waarop je de lokalisatie doet. Belangrijk is dat de volledige objectenboom die weergegeven gaat worden, is opgebouwd. In dit voorbeeld is er bovendien voor gekozen dat alle actie- en validatie-events zijn doorlopen en databinding is uitgevoerd; ofwel dat je zo dicht mogelijk op het moment van rendering zit - in het PreRender-event.

Een tweede aandachtspunt is het creëren van de resourcemanager voor de juiste assembly. Dit is belangrijk omdat hier de default-resources in zitten en op basis hiervan de satellite-assemblies worden opgezocht. ASP.NET maakt in runtime een class en assembly aan van de ASPX-pagina. Dit is dan ook het type van je huidige object, maar niet het juiste type voor lokalisatie. Voor lokalisatie heb je het type van de codebehind-class nodig, het BaseType van je ASPX-object.

Is het gebruik van de resourcemanager in ASP.NET te automatiseren?

Je kunt op generieke wijze de ASPX-objecthiërarchie doorlopen en voor iedere control de resourcemanager aanroepen en de gewenste eigenschappen lokaliseren.

In voorbeeldcode 3 zie je hoe de text-property van alle labels op de ASPX-pagina wordt gelokaliseerd. In de voorbeeld-applicatie behorend bij dit artikel kun je hiervan de volledige source-code vinden.

Hoe maak je een resourcefile?

Er zijn één of meer resourcefiles nodig. Deze resourcefile zul je zelf moeten maken. De meest voor de hand liggende

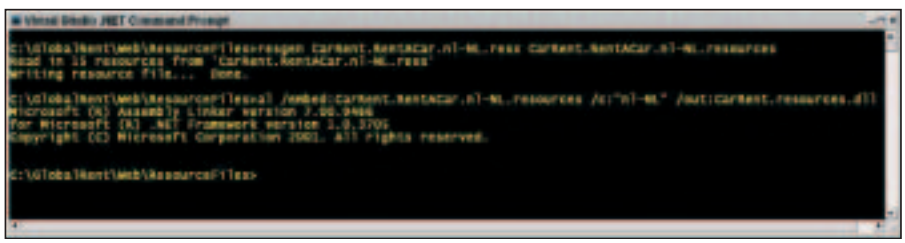
```
Private Sub Page_PreRender(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles MyBase.PreRender

  'Create resource manager....

  'Automatically process all the controls on the form
  IterateControls(Me)

End Sub

Private Sub IterateControls(ByVal ctrl As System.Web.UI.Control)
  Dim _ctrl As System.Web.UI.Control

  'Process the controls within the container
  If ctrl.HasControls()

    For Each _ctrl In ctrl.Controls
      ProcessControl(_ctrl)

      'If control is a container that holds controls loop again
      If (_ctrl.HasControls()) Then
        IterateControls(_ctrl)
      End If

    Next

  End If
End Sub

Private Sub ProcessControl(ByVal ctrl As System.Web.UI.Control)

  'Localize the text property when the control is a label
  If TypeOf ctrl Is System.Web.UI.WebControls.Label Then

    Dim _lblCtrl As System.Web.UI.WebControls.Label
    _lblCtrl = CType(ctrl, System.Web.UI.WebControls.Label)

    _lblCtrl.Text = ResourceManager.GetString(_lblCtrl.ID + ".Text")

    'etc

  End If

End Sub
```

Voorbeeldcode 3. Automatiseren van het aanroepen van de resourcemanager

keuze is een XML-resourcefile. Deze zijn eenvoudig te maken en muteren in Visual Studio.NET, zoals is te zien in afbeelding 4.

Afbeelding 4. Resourcefile in Visual Studio.NET

Deze resourcefile is de default resourcefile en moet dus in de main assembly worden opgenomen. Dit kun je doen door de file als een embedded resource toe te voegen aan je ASP-project, hierdoor ben je er ook zeker van dat je altijd een resourceset uitlevert. Tijdens het compileren worden de resources nu automatisch mee opgenomen in de main assembly. Naamgeving is hierbij cruciaal. De resources moeten voor het fallback-scenario dezelfde volledige naam hebben als de main assembly, aangevuld met de culture. De resourcefile uit afbeelding 3 is opgenomen als CarRent.RentACar.resources; dit kun je terugvinden in het assembly-manifest. Let wel op, voor jouw ASPX-pagina wordt automatisch een resourcefile aangemaakt. Deze zal bij compilatie dezelfde naam krijgen als jouw eigen resourcefile en dat geeft een conflict. Zorg er daarom voor dat je voor deze resourcefile als Build Action 'None' selecteert.



Afbeelding 5. Gebruik ResGen.exe en Al.exe

Voor de culture-specifieke resourcefiles doorloop je de volgende stappen:

1. Kopieer jouw default resourcefile voor iedere culture die je wilt ondersteunen en vertaal de waardes bij de items.
2. Genereer hiervan een binaire resourcefile met de tool ResGen.exe.
3. Compileer de binaire resourcefile tot een satellite-assembly met de tool Al.exe.

Het gebruik van de resourcegenerator-tool en de assemblylinker-tool zie je terug in afbeelding 5.

Is het aanmaken van resourcefiles in ASP.NET te automatiseren?

Het aanmaken van je default resourcefile kun je automatiseren. Je kunt dit niet tijdens designtime doen, aangezien de ASP.NET-controls alleen runtime bestaan. Door de ASPX-objecthiërarchie te doorlopen en voor iedere control de te lokaliseren eigenschappen weg te schrijven, creëer je de resourcefile.Ook de source-code hiervan vind je terug in de voorbeeldapplicatie behorend bij dit artikel. Deze voorbeeldapplicatie kun je downloaden van de websites www.microsoft.com of www.class-a.nl.

Welke resourcefile wordt gebruikt?

De resourcemanager kijkt standaard naar de culture van de aanroepende thread. Door de culture op deze thread in te stellen bepaal je dus welke resources worden gebruikt en hoe formattering plaatsvindt.

In voorbeeldcode 4 zie je dat er op de CurrentThread een CurrentCulture en een CurrentUICulture is. De CurrentUICulture wordt gebruikt door de resourcemanager. De CurrentCulture wordt gebruikt voor formattering. Deze bepaalt dus de weergave van getallen, data en bedragen. Hiervoor hoef je zelf niets te coderen.

.NET prima basis voor internationalisatie

.NET levert ons met de globalisatieservices een prima basis om onze applicaties te internationaliseren. We kunnen gebruik maken van een generiek mechanisme dat op basis van een culture uit de beschikbare resources de meest geschikte kiest. Doordat dit model is losgekoppeld van onze code kunnen de resources voor de verschillende culturen op ieder moment worden toegevoegd, aangepast of uitgebreid. Jammer genoeg is het bij het bouwen van webforms (nog) niet mogelijk om lokalisatie te implementeren met behulp van de .NET-designers. Dit is echter zeker geen reden om de .NET-infrastructuur niet te gebruiken. We kunnen natuurlijk altijd gewoon onze eigen tools bouwen!

```
Private Sub Page_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load
  'Create the culture info object
  Dim _culture As System.Globalization.CultureInfo
  _culture = New System.Globalization.CultureInfo(Me.cultureTextBox.Text)

  'Set the current culture for formatting of dates, numbers etc.
  System.Threading.Thread.CurrentThread.CurrentCulture = _culture

  'Set the current culture for resource localization
  System.Threading.Thread.CurrentThread.CurrentUICulture = _culture
End Sub
```

Voorbeeldcode 4. Instellen van de culture

Referenties

www.microsoft.com/globaldev
http://www.asp.net/Tutorials/quickstart.aspx
www.multilingual.com

Microsoft Global Development & Deployment Conference

18-20 FEBRUARI 2004. MICROSOFT CONFERENCE CENTER, REDMOND, WA, VS

De Microsoft® Global Development & Deployment Conference (GDDC) is de eerste Microsoft-conferentie die puur gericht is op de ontwikkeling en implementatie van internationale oplossingen die gebruikmaken van Microsoft-technologieën. De conferentie is van onschatbare waarde voor iedere ontwikkelaar of IT-professional die oplossingen ontwikkelt die meerdere talen en culturen ondersteunen. Door deze oplossingen kunnen ze sneller nieuwe markten toetreden en multinationals ondersteunen die communiceren in verschillende talen.

Voor wie is GDDC interessant?

Op de Microsoft GDDC wordt onderscheid gemaakt tussen ontwikkelaars die werken aan internationaal voorbereide .NET-, Win32- of weboplossingen en IT-professionals en systeembouwers die multinationals ondersteunen bij de communicatie in verschillende talen via Windows- en Office-technologieën.

Waarom moet ik GDDC bijwonen?

- Leer hoe u internationaal voorbereide oplossingen kunt ontwikkelen
- Leer hoe u internationale functies in Microsoft-producten kunt implementeren en beheren
- Treed toe tot nieuwe markten en verkrijg nieuwe klanten
- Ontmoet mensen die internationale oplossingen bouwen, ontwerpen of implementeren
- Een geweldige meerwaarde

Trajecten en inhoud:

De inhoud zal in het algemeen nieuw zijn. Sommige fundamentele onderdelen zoals de ontwikkeling van internationale Win32-toepassingen zijn eerder al elders gepresenteerd en zijn voor deze conferentie bijgewerkt. De belangrijkste trajecten zijn:

- Ontwikkelen van internationale Win32-toepassingen
- Internationale .NET-toepassingen ontwikkelen
- Internationale weboplossingen ontwikkelen
- Microsoft Office op internationale schaal inzetten en uitrollen
- Een netwerkinterface opbouwen die geschikt is voor meerder talen en culturen

Meer informatie over de conferentie:

www.microsoft.com/netherlands/msdn/events/gddc.asp