Wim Uyttersprot, Patrick Tisseghem, Gert Servranckx werken bij U2U

Het Information Bridge Framework 1.0

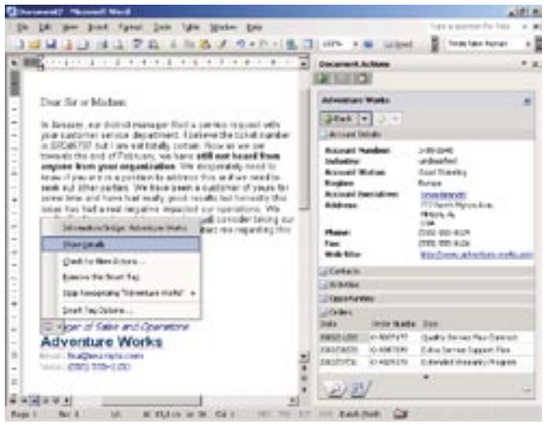
DE BRUG VAN MS OFFICE 2003 NAAR LINE-OF-BUSINESS-TOEPASSINGEN

Traditioneel is er een diepe kloof tussen de Line-Of-Business-systemen waarin operationele gegevens worden beheerd en de desktopomgeving met Microsoft Office die gebruikt wordt door informatiewerkers. Beide omgevingen hebben een gezamenlijk doel, met name de ondersteuning van de bedrijfsprocessen, maar hun invalshoek is verschillend. Het Microsoft Information Bridge Framework (IBF) ondersteunt ons om beide werelden te verbinden.

IBF is een nieuw framework dat speciaal is ontwikkeld voor de informatiewerker die gebruik maakt van een of meer Line-Of-Business LOB-toepassingen, maar gelijktijdig met Microsoft Office 2003 wil werken. IBF is vooral inzetbaar bij organisaties waar het nodig is informatiewerkers productiever te maken, door bedrijfstoepassingen en bedrijfsprocessen toegankelijk te maken vanuit Microsoft Office smart clients.

Situering IBF

Hoe worden vandaag in een bedrijfsomgeving de gegevens die binnenkomen via e-mail op een efficiënte manier verwerkt? Velen zullen antwoorden: door handmatig de data uit de mails te kopiëren naar de bedrijfstoepassingen. Nog steeds worden veel te vaak bedrijfsgegevens handmatig toegevoegd aan Excel-sheets en Word-documenten. Het Microsoft IBF legt een spreekwoordelijke brug tussen Office 2003 en de verschillende bedrijfstoepassingen die worden gebruikt door de informatiewerker. Microsoft heeft IBF ontworpen volgens een uitbreidbare, enterprise-scale architectuur waarop een nieuw soort toepassing kan worden ontwikkeld die LOB-systemen op een efficiënte wijze verbindt met de Office-ervaring. IBF staat toe om bedrijfsgegevens uit diverse LOB-systemen te consulteren, te analyseren en aan te passen vanuit Office smart clients. Met andere woorden: IBF zorgt ervoor dat businessinformatie en de bijhorende methoden in een enterprise-omgeving kunnen blijven gedefinieerd en door webservices aangeboden worden binnen de context van een Office-document. Informatiewerkers zijn op deze manier in staat hun bedrijfsgegevens efficiënt en centraal te beheren volgens procedures die zijn vastgelegd in bedrijfstoepassingen.



Afbeelding 1.Via een smart-tag in een Word-document legt IBF een link naar de toepassing die de corresponderende business-gegevens ophaalt en lob handelingen aanbiedt.

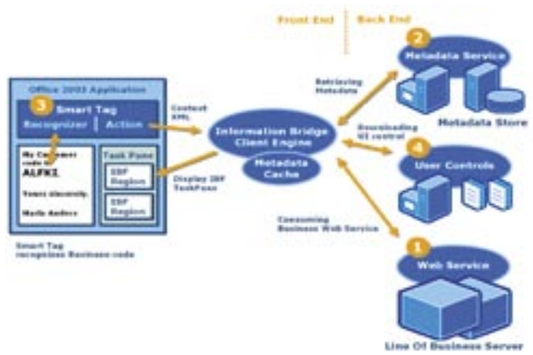
IBF versie 1.0 werkt met Word 2003, Excel 2003 en Outlook 2003. In versie 1.1 zullen bijkomende metadata-designers toegevoegd worden, en zal er support zijn voor InfoPath 2003. In latere versies wordt support verwacht voor SharePoint Portal Server Web Parts en uiteindelijk zal IBF ingebouwd worden in Windows 'Longhorn'. IBF biedt een gestandaardiseerde aanpak die is gestoeld op het definiëren van metadata. Ontwikkelaars van LOB-toepassingen ontwikkelen met name businessobjecten en bieden de door deze objecten geleverde diensten aan door middel van webservices. De solution-developers definiëren dan welke webservices op de smart client worden gebruikt. Ze combineren de businessobjecten en associëren deze met de user-interface van de Office-omgeving. Informatiewerkers gebruiken ten slotte de businessobjecten vanuit de context van hun Office-documenten en doen hierbij beroep op smart-tags, documenten die zijn verrijkt met XSD-schema's en een takenpaneel, de task pane.

Een typisch businessscenario

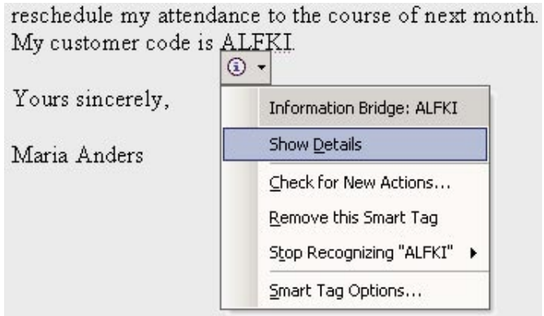
De kracht van IBF komt goed tevoorschijn in een typisch businessscenario: een informatiewerker wenst de businessgegevens te consulteren van een klant waarvan de firmanaam voorkomt in een Word 2003-document. In afbeelding 1 wordt de firmanaam "Adventure Works" als smart-tag herkend waardoor de informatiewerker met behulp van IBF de businessgegevens en services kan opvragen in de taakbalk.

Een IBF-toepassing ontwikkelen

Een typische IBF-toepassing ziet er vanuit het standpunt van de informatiewerker niet anders uit dan elk ander Word-, Excel- of Outlook-document dat een interactie heeft met de taakvenster.



Afbeelding 2. De onderdelen van een IBF-toepassing zijn de LOB-webservice, de metadata-service, de smart-tag en de UI-control.



Afbeelding 3. De klantcode ALFKI wordt herkend door de smart-tag waardoor IBF de bedrijfsinformatie in het taakvenster toont.

Inwendig bestaat de IBF-toepassing echter uit een aantal onderdelen op de desktop, een metadata-service op de IBF-server, en de webservices van een of meer LOB-toepassingen; zie afbeelding 2. Op de desktop vereist IBF als systeem een Microsoft Office 2003 Professional, Microsoft .NET Framework 1.1 en Windows 2000 Professional of hoger. De front-end componenten draaien binnen de Office 2003-toepassingen en bieden er de data aan die ze uitwisselen met de webservices op de server. De onderdelen zijn: de smart-tag-componenten, de userinterface-componenten onder andere voor het taakvensteren de Information Bridge client-engine. Deze client-engine haalt metadata op van de IBF-server, houdt deze bij in de cache op de client, en biedt geïnterpreteerde metadata aan in de Office-omgeving in de vorm van contextafhankelijke navigatie, menu's, businessobjecten en andere informatie-elementen.

Aan de serverzijde, draait de IBF metadata-service op een Windows Server 2003. De metadata op de server zijn opgeslagen in metadata-store die wordt beheerd door SQL Server 2000. De metadata beschrijven de views, acties, relaties en businessentiteiten van de IBF-toepassing op een gestandaardiseerde manier. Een IBF-toepassing ontwikkelen is in versie 1.0 nog niet zo eenvoudig. De IBF-ontwikkelomgeving is wel beschikbaar als een add-in in Visual Studio.NET, maar nog niet onmiddellijk als een RAD-tool. Het aantal designers en wizards is beperkt, wat betekent dat er nog steeds veel handmatig programmeer- en configuratiewerk bij komt kijken. De eerste IBF-toepassing die je zult ontwikkelen, zal met andere woorden behoorlijk wat scholing en planning vergen. Gebruikmakend van versie 1.0 kunnen we vijf hoofdstappen onderscheiden in het programmeerwerk dat nodig is om een IBF-toepassing te maken:

1. Maak een webservice aan conform de regels van het Information Bridge Framework
2. Creëer de IBF-metadata
3. Ontwikkel de user-interface die zal worden getoond in de IBF task pane
4. Definieer de smart-tags of een attached schema-document als ingangspunt voor IBF-operaties
5. Installeer en test de IBF-toepassing.

Voor de details van dit programmeerwerk verwijzen we naar het document "IBF in 5 steps, U2U tutor on the Information Bridge Framework" (). Om het concreet en vooral onmiddellijk toepasbaar te houden, wordt de IBF-technologie in dit voorbeeld toegepast op de bekende Northwind database van Microsoft SQL Server 2000.

De onderdelen van een IBF-toepassing

We bekijken de eenvoudige businesscase waarbij een informatiewerker zijn cursor plaatst op een klantcode in een Word-document en de corresponderende klantinformatie opvraagt in de task pane.

De informatiewerker wijst de klantcode ALFKI aan in een Word-document, waardoor er een smart-tag verschijnt. Selecteert hij het menu 'Show Details' dan heeft dit tot gevolg dat in het taakvester de bedrijfsinformatie van Afreds Futterkiste verschijnt; zie afbeelding 3.

Wat gebeurt er nu achter de schermen? Welke informatie wordt hier op welke manier uitgewisseld? We bespreken de verschillende onderdelen van de IBF-toepassing en beschrijven de weg die wordt gevolgd vanaf het herkennen van de klantcode als een smart-tag in Word tot de weergave van de klantinformatie in de task pane.

Onderdeel 1: de webservice conform IBF

IBF-clienten halen hun businessdata op via webservices. Deze webservices moeten conform de richtlijnen van het Information Bridge Framework zijn (codevoorbeeld 1), wat onder andere betekent dat hun definities moeten overeenkomen met schema's in de IBF metadata-store. In onze businesscase maken we gebruik van de Customers-webservice met een GetCustomerInfo-webmethode. Als input geven we een klantcode op om de klantinformatie als output te verkrijgen. De richtlijnen van IBF die zijn toegepast op ons voorbeeld verplichten ons om de klantcode te encapsuleren in een reference-type – in dit geval de CustomerReference-class; codevoorbeeld 2. Bovendien moet de klantinformatie geëncapsuleerd zijn in een view-type – de CustomerView-class; codevoorbeeld 3.

Onderdeel 2: de metadata-store op de IBF Server

De metadata-store in een IBF-toepassing bevat alle informatie over de businessdata die nodig is om een vraag te beantwoorden komende van de IBF-client, bijvoorbeeld een smart-tag. Zo beschrijven de metadata onder andere de structuur van de businessdata (schema's), waar de businessdata zich bevinden (ViewLocators), wat er met de businessdata kan gebeuren (actions), en

```
[WebService(Namespace="urn:schemas-u2u-net/Customers")]
public class Customers : System.Web.Services.WebService
{
   [WebMethod(Description="Retrieve info for a specific customer")]
   public CustomerView GetCustomerInfo(CustomerReference customerRef)
   {
      ...
   }
}
```

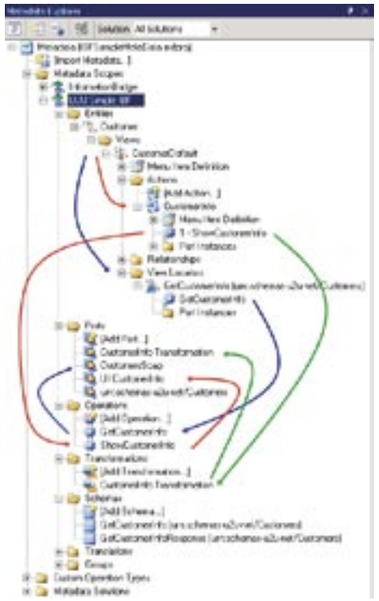
Codevoorbeeld 1.
Webservices moeten conform zijn aan de richtlijnen van IBF

```
[XmlRoot("CustomerReference", Namespace="urn:schemas-u2u-net/Customers")]
public class CustomerReference
{
   [XmlAttribute]
   public string CustomerID;
}
```

Codevoorbeeld 2.
Het reference-type (input) conform de richtlijnen van IBF

```
[XmlRoot("CustomerView", Namespace="urn:schemas-u2u-net/Customers")]
public class CustomerView
{
   [XmlAttribute]
   public string CustomerID;
   XmlElement]
   public string CompanyName;
   [XmlElement]
   public string Address;
   ...
}
```

Codevoorbeeld 3.
Het view-type (output) conform de richtlijnen van IBF



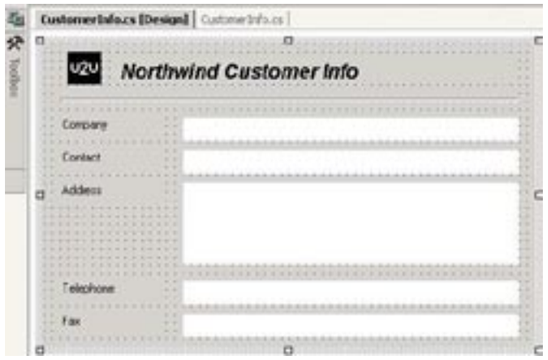
Afbeelding 4. De Metadata Explorer

hoe de businessdata moeten worden getoond (transformaties). De metadata-store wordt automatisch als een SQL Server-database geïnstalleerd tijdens de set-up van IBF. Het beheer van de metadata-store gebeurt in Visual Studio.NET via een speciale IBF project-template en een add-in, met name de Metadata Explorer; afbeelding 3. Zowel de IBF-template als de Metadata Explorer worden automatisch geïnstalleerd tijdens de set-up van IBF.
Metadata kunnen worden uitgewisseld tussen de metadata-store en het metadata-project via de metadata-webservice bijvoorbeeld http://U2Ustore:8082/IBFWriteService.asmx zoals te zien in afbeelding 5.

Metadata toevoegen aan het project kan op verschillende manieren, handmatig door de achterliggende xml-file te redigeren bij Entities, Views, en Viewlocators, via wizards (Actions, Ports, Operations, Schema's...), en via import uit xml-files. De Metadata Explorer laat toe de belangrijke metadata-elementen te beheren en te redigeren. Belangrijke elementen zijn:

- Ports: de poorten naar de webservice die de SOAP-details abstraheren, en de poorten naar de XSD schema-files voor de input- en output-types.

Afbeelding 5. Importeren van metadata gebeurt van de metadata-store naar het metadata-project; publiceren van metadata gebeurt richting store.



Afbeelding 6. De IBF user-interface wordt gedefinieerd als een Windows control in .NET

- Schemas: de schema's van de IBF webservice in apart request en response message-formaat.
- Operations: de methoden die door IBF aanroepen kunnen worden.
- Entities: de data-entiteiten en hun views die door IBF gebruikt moeten worden.

Onderdeel 3: De smart-tag component of het 'attached schemadocument'.
Als informatiewerkers met businessdata omgaan, gebeurt dit vanuit de tekst in een Office-document. Een smart-tag is een object dat zich 'at runtime' creëert en nestelt in het Office-document. Door toedoen van de smart-tag-technologie wordt het Office-document mogelijk onderdeel van de IBF-toepassing. Of informatiewerkers nu een bestaand document consulteren, of een nieuw document aanmaken, de smart-tag legt de link van het Office-document naar de IBF-toepassing. Een smart-tag wordt gedefinieerd door een smart-tag-component, dat in .NET typisch wordt aangemaakt als een class library object. In plaats van gebruik te maken van smart-tags laat IBF ook toe dat we werken met schema's die een vaste structuur opleggen aan een Word- of Excel-document om dan vanuit de 'gemapte' xml-velden de contextinformatie door te geen aan de IBF client-engine.

Onderdeel 4: De IBF user-interface
Een wezenlijk onderdeel van een IBF-toepassing is de user-interface in de Office-omgeving. In Word en Excel gebruiken de informatiewerkers hiervoor de task pane. In Outlook wordt deze zelf het taakvenster getoond als een top level window. De user-interface wordt in ons voorbeeld aangemaakt in .NET als een Windows user control, maar IBF ondersteunt ook gewone HTML, ingebouwde IBF-controls en de transformatietaal XSL voor de definitie van de interface binnen de task pane.

IBF in werking

We volgen nu de weg door de IBF-functionaliteit vanaf het moment dat een term herkend wordt als smart-tag in Word tot aan het tonen van de business data in de task pane.

Stap 1: De businesscode wordt herkend
We beginnen bij de informatiewerker die een tekst typt in Word 2003 waarin de term ALFKI voorkomt. ALFKI wordt door de smart-tag engine van Office herkend als klantcode. Achter de schermen werd de tekst namelijk gescand door de SmartTagRecognizer van onze smart-tag; zie codevoorbeeld 4. De smart-tag is operationeel in Office en vergelijkt elk woord uit de tekst met de codes in een XML-bestand genaamd SmartTagTerms.xml. Onze smart-tag-component herkent ALFKI als klantcode, omdat deze code is opgenomen in dit XML-bestand; zie codevoorbeeld 5.

Stap 2: De request wordt verwerkt door de IBF client-engine
Klikt de informatiewerker op 'Show Details' in het contextmenu van de smart-tag, dan stuurt de Execute-methode van het Action-

```
public class SmartTagRecognizer : ...
{
  void Recognizer2(...)
  {...
    switch (node.Name)
    {
      ...
      case "Customer":
        contextString = CustomerContextXml;
        break;
    }
  ...
  }
  ...
}
```

Codevoorbeeld 4.
De SmartTagRecognizer

```
<Terms>
  <Customer>ALFKI</Customer>
  <Customer>Anton</Customer>
</Terms>
```

Codevoorbeeld 5.
SmartTagTerm s.xml bevat de Customer-code ALFKI

```
<?xml version="1.0"?>
<ContextInformation xmlns:xsd="http://www.w3.org/2001/XMLSchema
  mlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  MetadataScopeName=U2U Simple IBF
  EntityName=Customer ViewName=CustomerDefault
  xmlns=http://schemas.microsoft.com/
      InformationBridge/2004/ContextInformtion>
  <Reference>
    <GetCustomerInfo xmlns=urn:schemas-u2u-net/Customers>
        <CustomerReference CustomerID={0}
        iwb:MetadataScopeName=U2U Simple IBF
        xmlns:iwb=http://schemas.microsoft.com/InformationBridge/2004
        iwb:EntityName=Customer iwb:ViewName=CustomerDefault />
    </GetCustomerInfo>
  </Reference>
</ContextInformation>
```

Codevoorbeeld 6.
De XML-message die door de IBF-client-engine naar de IBF-server wordt gestuurd.

object in de smart-tag een XML-message door naar de IBF client-engine; zie codevoorbeeld 6. De engine pakt de boodschap uit om deze verder te bewerken op basis van de metadata die werden opgehaald uit de metadata-repositoy.

Het formaat van deze XML-message wordt bepaald door de metadata op de IBF-server, met name het schema GetCustomerInfo, dat verwijst naar de poort urn:schemas-u2u-net/Customers. De XML-message bevat een request GetCustomerInfo met als argument een CustomerReference die de CustomerID bevat. De request verwijst naar de entiteit Customer en de view CustomerDefault. In wezen vraagt de XML-message dus om de Customer-gegevens van ALFKI te tonen volgens een CustomerDefault-view.

Stap 3: Het ophalen van de benodigde metadata door de client-engine
De XML-contextmessage wordt beantwoord door de IBF client-engine waar de CustomerDefault-view in de metadata vanaf nu het coördinerende werk overneemt binnen de omkadering van het IBF. Het antwoord dat door de IBF client-engine wordt aangeleverd, wordt geformuleerd op basis van de metadata die wordt opgehaald en gecached via het aanspreken van de metadata-webservices die worden aangeboden door de IBF-servercomponent. De metadata bestaan in ons voorbeeld uit drie onderdelen: de url van de business-webservice, de url van de .NET-assembly die de usercontrol bevat en de naam van de usercontrol-class. Met behulp van de Metadata Explorer vinden we de verschillende onderdelen.

Onderdeel 1: de url van de business-webservice
De CustomerDefault-View gaat naar de ViewLocators en vindt daar de operatie GetCustomerInfo die moet worden uitgevoerd om de Customer-data op te halen. De operatie GetCustomerInfo verwijst via een property naar de poort CustomerSoap, die het eerste onderdeel van het antwoord bevat. Dit is het voorbeeld de locatie van de business-webservice: http://www.u2u.net/CustomerServices/Customer.asmx

Onderdeel 2: url van de .NET-assembly
De CustomerDefault-View gaat naar de Actions en vindt daar de actie CustomerInfo. Deze actie verwijst naar de operatie ShowCustomerInfo. In de Operations verwijst ShowCustomerInfo via een property naar de poort UI CustomerInfo. Bij de Ports vermeldt de poort UI CustomerInfo via een property naar de url van de Windows Control-DLL, in het voorbeeld: IBFNorthwindUserControls.dll.

Onderdeel 3: de naam van de usercontrol-class
Eén van de properties van de ShowCustomerInfo-operatie in de Actions, met name de TransFormationInstances-property verwijst naar de transformatie 'CustomerInfo Transformation'. Bij de Transformations verwijst de 'CustomerInfo Transformation' via een property naar de poort CustomerInfo Transformation. Bij Ports vinden we deze poort waarvan de Data-property de XSL weergegeven in codevoorbeeld 7:

Deze XSL bevat een link naar de usercontrol-class met de naam 'IBFNorthwindUserControls.CustomerInfo', die de class is die zal

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
              xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <msibf:Region
      xmlns:msibf="http://schemas.microsoft.com/InformationBridge/2004"
          Enabled="true">
      <msibf:RegionProperties RegionName="RegionCustomerInfo"
          Caption="Customer Info" Description="Customer Info"
          TypeName="IBFNorthwindUserControls.CustomerInfo"
          ShowAs="ExpandedRegion">
        <xsl:copy-of select="/" />
      </msibf:RegionProperties>
    </msibf:Region>
  </xsl:template>
</xsl:stylesheet>
```

Codevoorbeeld 7.
De XSL-property die verwijst naar de user-interface

```
public class CustomerInfo : UserControl, IRegion
{
  ...
  public XmlNode Data
  {
    set
    {
      labelCompany.Text = value.FirstChild.ChildNodes[0].InnerText;
      labelAddress.Text = value.FirstChild.ChildNodes[2].InnerText +
      ...
    }
  }
}
```

Codevoorbeeld 8.
De Data-property van de usercontrol zorgt voor het daadwerkelijk invullen van de klantgegevens in de UI-controls

worden gebruikt en getoond in een region die deel uitmaakt van de IBF user-interface; zie afbeelding 6.

De uiteindelijke response die wordt gegeven door de IBF-meta-data-service bestaat dus uit:

- 'http://www.u2u.net/CustomerServices/Customer.asmx' als url van de business-webservice
- 'IBFNorthwindUserControls.dll' als url van de .NET assembly
- 'IBFNorthwindUserControls.Cus tomerInf' als naam van de usercontrol-class

Stap 4: De respons wordt verwerkt door de IBF client-engine
De IBF client-engine zorgt nu voor het daadwerkelijk aanroepen van de business-webservice, en het tonen van de hieruit verkregen data in de IBF task pane. De user control ontvangt de busines-sdata via de IRegion.Data-property. Deze data zijn geformatteerd als XmlNode conform het GetCustomerInfoResponse-schema. De Data-property van de user control zorgt voor het daadwerkelijk invullen van de klantgegevens in de UI controls; codevoorbeeld 8.

Voor de businesscase die we in dit artikel bespraken is het gewens-te resultaat behaald, maar in businesscases waar data moeten wor-den onderhouden, of waar businessactiviteiten gestart moeten wor-den, is er nog wat bijkomend - weliswaar analoog - werk te doen.

Referenties

MSDN over het Information Bridge Framework:
http://msdn.microsoft.com/office/understanding/ibframework/default.aspx
"IBF in 5 steps, U2U tutor on the Information Bridge Framework"
http://www.u2u.net/ibf

Wim Uyttersprot (wim@u2u.be) en **Patrick Tisseghem** (patrick@u2U.be) zijn managing partners van U2U. **Gert Servranckx** (gert@u2u.be) werkt bij U2U als .NET trainer. (www.u2u.net).