Interoperabiliteit in Forms met .NET

JNBridgePro is een krachtige oplossing

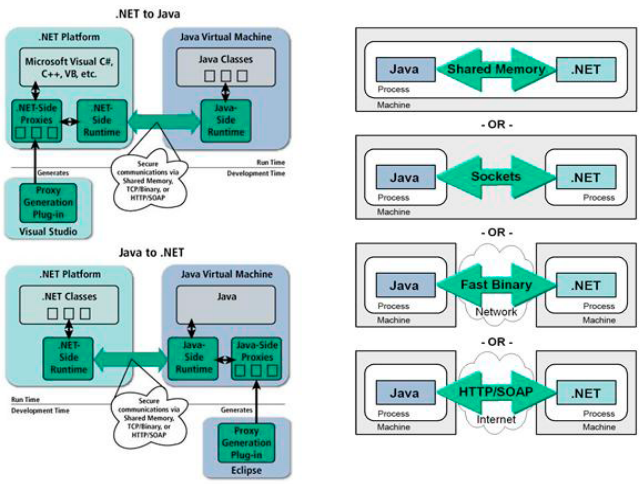
Oracle Forms wordt gebruikt voor veel front-end applicaties. Stel nu dat we vanuit een Oracle Forms front-end gebruik willen maken van de functionaliteit uit een .NET back-end applicatie. Hoe kunnen we dit voor elkaar krijgen?

Oracle Forms kan gebruik maken van Java-klassen. Op deze manier is het mogelijk een Forms front-end applicatie aan te passen naar je eigen wensen en extra functionaliteit toe te voegen. Nu is er een product van JNBridge, genaamd JNBridgePro, dat het mogelijk maakt om vanuit een Java-klasse, die in de Forms front-end zit, de .NET back-end code te gebruiken. Natuurlijk kun je zelf aan de slag met de Java Native Interface, maar dit is in verhouding met JNBridgePro erg veel werk. JNBridgePro is een Java en .NET interoperabiliteitsproduct dat het toelaat om toegang te krijgen tot de gehele objectgeoriënteerde API van de andere kant (Java of .NET) in hetzelfde proces of over een netwerk. Zo is er communicatie mogelijk van een Java API met een .NET API en vice versa. Dit brengt een groot aantal voordelen met zich mee zoals:

- Bi-directioneel, high-performance Java .NET interoperabiliteit;
- Toegang tot Java-klassen in .NET alsof het .NET klassen zijn;
- Toegang tot .NET-klassen in Java alsof het Java klassen zijn;
- Toegang tot alles over de platformgrens heen zoals; objecten, klassen, instance variabelen, static variabelen, velden, eigenschappen en methodes.

Dit artikel verduidelijkt de architectuur van JNBridgePro en laat aan de hand van een voorbeeldimplementatie zien hoe JNBridgePro kan worden gebruikt. Uiteraard is dit voorbeeld maar een topje van de ijsberg aan mogelijkheden waarover JNBridgePro beschikt.

De communicatie tussen Java en .NET kan lopen via verschillende kanalen en er zijn diverse implementatiemogelijkheden. Figuur 1 geeft aan de linkerkant de implementaties weer van JNBridgePro en aan de rechterkant van het figuur zijn de communicatiekanalen weergegeven.



Figuur 1: Implementatie JNBridgePro plus communicatiekanalen.

Met deze implementaties is het mogelijk om een applicatie te bouwen die communiceert met een andere applicatie:

- Op dezelfde machine in hetzelfde proces (embedded) via gedeeld geheugen (shared memory);
- Op dezelfde machine in verschillende processen via socket communicatie;
- Over een netwerk met een binair protocol;
- Over het internet via HTTP of SOAP.

Mocht het noodzakelijk zijn dat er toch gewisseld moet worden van communicatiekanaal, dan moeten alleen de configuratieparameters worden veranderd en is het niet nodig aanroepende code te veranderen.

JNBridgePro creëert de interoperabiliteitsbrug door het genereren van proxies. Deze proxies dienen gegenereerd te worden via de bijgeleverde plugins voor Visual Studio en Eclipse of door middel van een standalone (jnbproxyGui)- of CLI-tool. Via deze plugin of tool kun je zelf kiezen welke klassen er zichtbaar moeten zijn voor de buitenwereld. De tool/plugin zoekt zelf, waar nodig, de ondersteunde klassen erbij. De .NET-kant maakt

gebruik van gecompileerde Java-bestanden of JARs om de proxies te genereren, broncode is dus niet nodig. Voor de Java-kant wordt er gebruik gemaakt van de DLL bestanden die .NET creëert. Tijdens het genereren van de proxies voegt de tool/plugin zelf code toe om te communiceren met de JNBridgePro API.

Zodra de proxies zijn gegenereerd kun je via deze proxies toegang krijgen tot de onderliggende Java-klassen vanuit .NET (of .NET-klassen vanuit Java). Een proxyklasse is in feite een JNBridgePro aanroep waarbinnen onder andere gegevensconversie plaatsvindt en een instantie van een object van het andere platform wordt aangeroepen.
De .NET-code draait in een .NET CLR (Common Language Runtime) en de Java-code wordt uitgevoerd in een JVM.

Naast het generen van proxies zorgt JNBridgePro ook voor:
- Cross-platform exceptie afhandeling;
- Geïntegreerde object Lifecycle Management;
- Garbage collection;
- Gegevenstype conversies;
- Het marshalling en unmarshalling van objecten;
- Referentie- en communicatie management;
- Thread safety tussen de platformen;
- Transactiemanagement.

JNBridgePro biedt ook ondersteuning voor het kunnen doorgeven van objectreferenties in plaats van objectwaardes waardoor overhead wordt beperkt. Verder is Eventhandling transparant, het is dus mogelijk een Java Eventhandler te koppelen aan een .NET-klasse en een .NET Eventhandler aan een Java-klasse. In de volgende alinea's wordt een voorbeeldimplementatie uiteengezet om een werkende koppeling tot stand te brengen tussen een .NET-applicatie en een Oracle Forms applicatie via een Java-klasse. Voor de communicatie is gekozen voor het gebruik van sockets, vanuit verschillende processen op dezelfde machine.

De .NET back-end

Als eerste hebben we een simpele .NET back-end applicatie nodig. Deze .NET-applicatie bestaat uit een klasse die het benodigde DLL-bestand laadt waarin de Form1-klasse zit. Form1 bevat de businesslogica en user interface van de applicatie. De namespace en klassendefinitie hieronder zorgen voor de initialisatie van de System DLL-bestanden en het starten van de .NET-kant. De klasse opent een consolescherm waardoor de .NET-applicatie actief blijft, totdat de gebruiker met ENTER aangeeft de applicatie te willen stoppen.

```
using System;
using com.jnbridge.jnbcore;
```

```
namespace SwingInteropDotNetSide
{
          class DotNetSide
          {
                    [STAThread]
                    static void Main(string[] args)
                    {
       // specify the assemblies we'll be accessing from Java
                              string[] assemblies = {
          "SwingInterop.dll",
"System.Windows.Forms, Version=1.0.5000.0, Culture=neutral,
PublicKeyTok en=b77a5c561934e089"
                              };
  // need to initialize DotNetSide with the names of all the assemblies
  // from which you will be referencing classes
          DotNetSide.startDotNetSide(assemblies);
          Console.WriteLine("Hit <return> to exit");
          Console.ReadLine();
          DotNetSide.stopDotNetSide();    // end it (optional)
                    }
          }
}
```

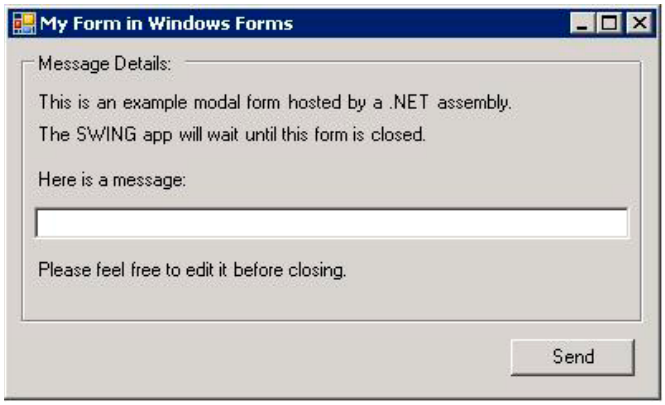
In de SwingerInterop.dll zit de Form-subklasse genaamd Form1. Dit is een simpel dialoogscherm. De belangrijkste functionaliteit hierin is wat er gebeurt als er op de 'Send' knop geklikt wordt:

```
namespace SwingInterop
{
          public class Form1 : System.Windows.Forms.Form
          {

                    public String Message
     {
                    get { return message; }
         set
         {
          message = value;
          this.textBox1.Text = message;
                              }
                    }
                    …
private void send_Click(object sender, System.EventArgs e)
                    {
if (buttonClick != null)
{
buttonClick.DynamicInvoke(new object[]{ this, new
JavaWindowEventArgs(this.textBox1.Text) } );
}
this.Message = this.textBox1.Text;
                    }
          }
          public class JavaWindowEventArgs : EventArgs
          {
                    public string message;

                    public JavaWindowEventArgs(string theMessage):
base()
                    {
                              message = theMessage;
                    }
          }
          …
}
```

De JNBridgePro API zorgt voor de verdere afhandeling mits de goede configuratie is toegepast. Omdat in dit voorbeeld is gekozen voor socket communicatie is het van belang aan te geven waar de twee zijdes, .NET en Java, zich bevinden:

```
<jnbridge>
    <dotNetToJavaConfig scheme="jtcp" host="localhost" port="8085"/>
    <javaToDotNetConfig scheme="jtcp" port="8086"/>
 </jnbridge>
```

Deze configuratie moet gedaan worden in het .NET applicatie configuratiebestand in het .NET-project. Het scherm behorende bij de Form1-klasse is weergegeven in figuur 2.



Figuur 2: Form1 window.

De tussenlaag: Java

Voor de Java-kant moet er een klasse worden gemaakt, deze noemen we JavaDotNet. Deze klasse draait achter het opstartscherm van de Oracle Forms applicatie, zodat deze de gehele sessie actief blijft. Wanneer deze JavaDotNet instantie wordt geïnitialiseerd opent deze het .NET Form1 window (de .NET applicatie) in een aparte thread met de juiste instellingen. Om ervoor te zorgen dat Oracle Forms deze JavaDotNet-klasse kan gebruiken dient deze een subklasse te zijn van de door Forms aangeleverde Java-klasse genaamd VBean.
Oracle Forms roept de init methode aan nadat de JavaDotNet-klasse geïnstantieerd is.

```
public class JavaDotNet extends VBean implements EventHandler,
AsyncCallback{
.....
public void init(IHandler handler) {
                    mHandler = handler;
                    super.init(handler);
Properties prop = new Properties();
// Java-side (.NET-to-Java) properties
          prop.setProperty("javaSide.serverType", "tcp");
          prop.setProperty("javaSide.workers", "5");
          prop.setProperty("javaSide.timeout", "10000");
          prop.setProperty("javaSide.port", "8085");
          // .NET-side (Java-to-.NET) properties
          prop.setProperty("dotNetSide.serverType", "tcp");
          prop.setProperty("dotNetSide.host", "localhost");
          prop.setProperty("dotNetSide.port", "8086");
          DotNetSide.init(prop);
          form1 = new Form1();
          form1.add_ButtonClicked(this);
          worker = new MyWorker();
          worker.start();
          }
.....
```

```
public static class MyWorker extends Thread
          {
                    public void run()
                    {
                              form1.ShowDialog();
                              form1.Dispose();
                    }
          }
}
```

De klasse MyWorker is een static klasse die het Form1 daadwerkelijk opent. De reden dat dit in een aparte thread moet gebeuren is dat op deze manier de applicatie open blijft staan en de Java-kant niet blijft wachten op antwoord van het Form1, maar door kan gaan met het uitvoeren van de code. De regel form1.ShowDialog(); in de run() methode van MyWorker blijft altijd draaien. Er wordt pas een regel code uitgevoerd na de ShowDialog() regel als het Form1 niet meer actief, dus afgesloten, is.

De geopende .NET-applicatie bevat een scherm, zoals getoond in figuur 2, dat tekst kan ontvangen vanaf de Java-kant en tekst terug kan sturen naar de Java-kant. Form1 in de .NET-applicatie wordt meteen geopend zodra de init() methode, en dus de run() in MyWorker, wordt aangeroepen. Indien het Form1 afgesloten wordt, is er in dit voorbeeld geen communicatie meer mogelijk. Vanuit de Oracle Forms applicatie is het mogelijk de JavaDotNet-klasse aan te roepen. Dit gebeurt door een vooraf gedefinieerde property te setten in de JavaDotNet-klasse vanuit PL/SQL (Oracle Forms gedeelte en bijbehorende PL/SQL code wordt later besproken).

```
public class JavaDotNet extends VBean implements EventHandler,
AsyncCallback{
.....
// properties you can set
protected static final ID SET_MESSAGE =
ID.registerProperty("SET_MESSAGE ");
// properties you can get
protected static final ID GET_MESSAGE =
ID.registerProperty("GET_ MESSAGE ");
// events you can raise
protected static final ID EVT_MESSAGE_BACK = ID.registerProperty("EVT_
MESSAGE_BACK ");

public boolean setProperty(ID property, java.lang.Object value) {
          if (property == SET_ MESSAGE) {
                    form1.Set_Message((String) value);
                    return true;
          } else //Default behavior.
          {
                    return super.setProperty(property, value);
          }
}
.....
}
```

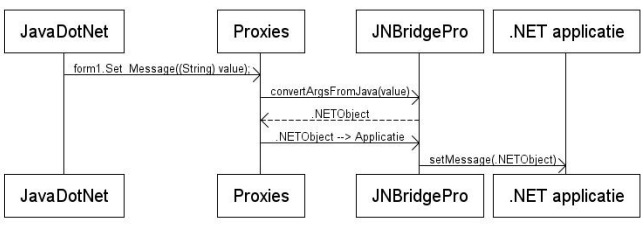
De form1.Set_message is een aanroep op een geproxied .NET-object waarmee tekst naar de .NET Form1 instantie wordt verstuurd. De methode geeft een returnwaarde 'true' om aan te geven dat het setten van de property is gelukt. Omdat wij

niet geïnteresseerd zijn in andere dan de SET_MESSAGE property, wordt de rest doorgegeven aan de superklasse.
Er is in de JavaDotNet-klasse een listener geïmplementeerd vanuit de JNBridgePro API die 'luistert' naar berichten van het Form1. Tevens implementeert de JavaDotNet-klasse een System.EventHandler (.NET EventHandler), zodat deze kan worden geregisterd bij de .NET Form1 instantie zoals gebeurde in de init() methode met form1.add_ButtonClicked(this).

```
import SwingInterop.Form1;
import SwingInterop.JavaWindowEventArgs;
import System.EventArgs;
import System.EventHandler;
import System.Object;

import com.jnbridge.jnbcore.AsyncCallback;
public class JavaDotNet extends VBean implements EventHandler,
AsyncCallback{
.....
          // Deze methode is een implementatie van de .NET EventHandler
interface
public void Invoke(Object sender, EventArgs args)
{
          JavaWindowEventArgs javaArgs = (JavaWindowEventArgs) args;
          output = javaArgs.Get_message();
          dispatch_event(EVT_MESSAGE_BACK);
}

public void dispatch_event(ID id) {
CustomEvent ce = new CustomEvent(mHandler, id);
          dispatchCustomEvent(ce);
}
.....
}
```

Op het moment dat er een bericht wordt ontvangen, omdat de gebruiker in het .NET Form1 window een message heeft ingegeven en op send heeft gedrukt, maakt de JavaDotNet-instantie in de methode Invoke() een CustomEvent aan voor Oracle Forms. Hierop kan worden gereageerd met behulp van PL/SQL-code in de Oracle Forms applicatie door de waarde op te halen door het aanroepen van de getProperty methode met als property ID GET_MESSAGE . De daadwerkelijke PL/SQL-code om dit te doen komt later aan bod. Ook hier geldt dat wij niet geïnteresseerd zijn in andere dan de GET_MESSAGE property, de rest wordt doorgegeven aan de superklasse.

```
public java.lang.Object getProperty(ID property) {
          if (property == GET_MESSAGE) {
                    // return the corresponding value
                    return output;
          } else // default behaviour
          {
                    return super.getProperty(property);
          }
}
```



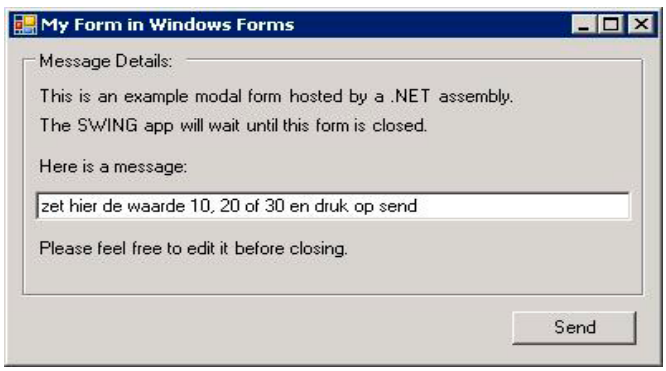
Figuur 3: Schematische weergave van de communicatie tussen de JavaDotNet-klasse en .NET-applicatie.

Deze communicatie blijft doorgaan totdat een van de twee wordt afgesloten.
Om de communicatie tussen de JavaDotNet-klasse met de proxies en met JNBridgePro te verduidelijken is in figuur 3 schematisch weergegeven wat er gebeurt indien de Set_Message() methode wordt uitgevoerd vanuit de JavaDotNet-klasse op het Form1 Object.

De front-end in Oracle Forms

Voor de forms front-end maken we gebruik van een demo-applicatie van Headstart in combinatie met Jinitiator 1.3.1.18. In het startform van deze forms front-end applicatie is een bean-item opgenomen met als implementatieklasse de klasse van het eerder beschreven Java-component, de JavaDotNet-klasse. Aangezien het startform wordt opgestart bij het starten van de forms front-end applicatie en deze altijd 'actief' blijft, hoeft de JavaDotNet-instantie alleen in het startform opgenomen te worden. Aangezien de JavaDotNet-klasse meteen wordt geïnstantieerd, verschijnt direct de .NET Form1 window.
Om aan te tonen dat het mogelijk is vanuit de forms front-end via Java naar .NET een boodschap te sturen is er in het startform een extra knop opgenomen. Deze knop stuurt een tekst naar het .NET Form1 instantie (via de JavaDotNet-klasse), waarin deze vervolgens wordt getoond (zie figuur 4). De procedure die wordt uitgevoerd bij het klikken op de knop roept de javabean aan en vervolgens de JavaDotNet-instantie en ziet er als volgt uit:

```
begin
set_custom_property ('START.MIJNBEAN'
 , 1
          , 'SET_MESSAGE'
```



Figuur 4: Form1 window met boodschap vanuit de procedure in de forms front-end applicatie.

```
 , 'zet hier de waarde 10, 20 of 30 en druk op send'
);
end;
```

In het Form1 window kan een code 10, 20 of 30 worden getypt en verzonden richting de forms front-end applicatie. Deze codes stellen departmentnummers voor.
Aan de hand van een code wordt er genavigeerd naar een Employee Form in de forms front-end applicatie waarin de employee's van dat departmentnummer getoond worden. Eerder hebben we gezien dat in de invoke methode van de JavaDotNet-klasse, wanneer op de Send knop wordt gedrukt, een EVT_MESSAGE_BACK event wordt gedispatched. In de WHEN-CUSTOM-ITEM-EVENT trigger in de forms front-end applicatie wordt dit event afgevangen. Door de GET_MESSAGE property vervolgens uit de JavaDotNet-instantie te lezen kan het volgende scherm worden opgestart, waarbij, met de zojuist opgehaalde waarde, een query wordt uitgevoerd (zie figuur 5 en 6).

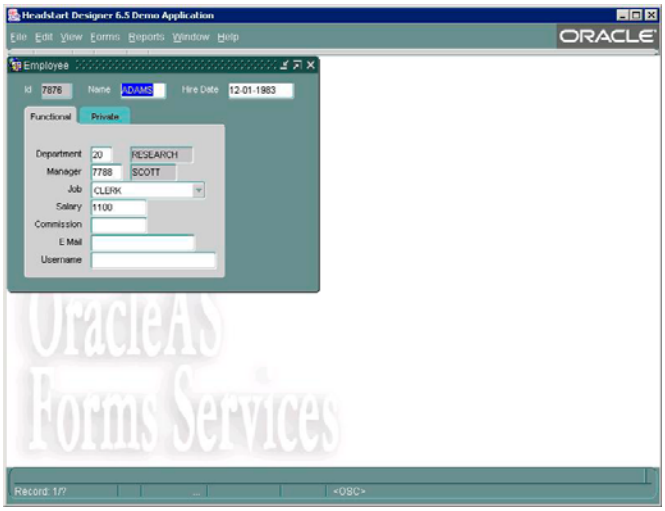
Figuur 5: een ingevuld WinForm.


```
declare
eventname varchar2 (20);
argList paramlist;
returnValue varchar2 (255);
begin
eventname := :system.custom_item_event;
if eventname = 'EVT_MESSAGE_BACK'
then
--
-- sluit alle forms behalve de startform
--
qms$application.close_all_forms;
--
-- haal de ingevulde waarde op uit de .JavaDotNet instantie
--
returnValue := get_custom_property ('START.MIJNBEAN', 1, 'GET_MESSAGE');
--
-- bouw parameterlijst op voor de aan te roepen form
--
argList := get_parameter_list ('HSD0004F');
if not id_null (argList)
then
destroy_parameter_list (argList);
end if;
argList := create_parameter_list ('HSD0004F');
add_parameter (argList, 'P_DEPTNO', text_parameter, returnValue);
add_parameter (argList, 'CG$STARTUP_MODE', text_parameter, 'AUTO
QUERY');
--
-- roep de tweede form aan
--
call_form ( 'hsd0004f'
, hide
, no_replace
, no_query_only
, argList
);
--
-- zet de uitgeschakelde melding weer aan
--
qms$forms_errors.delete_suppress_message ('FRM-41351');
end if;
end;
```

Figuur 6 toont het resultaat van het versturen van een message vanuit het .NET Form1 window, de JavaDotNet-klasse en de forms front-end applicatie.

Opletten

Omdat dingen niet zomaar out of the box werken, zijn er een aantal punten waar goed opgelet dient te worden.
De .NET-kant moet actief zijn anders is het niet mogelijk het



Figuur 6: Resultaat van de navigatie en het uitvoeren van de query.

WinForm te openen. Indien de .NET-kant niet actief is, kun je de volgende melding krijgen:

Problem opening the client transport mechanism.Did you start the .NET side?
Inner exception:
com.jnbridge.jnbcore.clientTransports.g: problem opening the socket. Did you start the .NET side?

Omdat er wordt gewerkt met socket verbindingen moeten de volgende instellingen worden gedaan in de java.policy file van de Jinitiator:

```
permission java.lang.RuntimePermission "*", "accessDeclaredMembers";
permission java.net.SocketPermission "*", "accept,connect,listen,resolve
";
permission java.lang.reflect.ReflectPermission "*", "suppressAccess

Checks";
```

Indien dit niet wordt aangepast kan er een AccessControlException worden gegooid.
Enkele voorbeelden hiervan zijn;

java.security.AccessControlException: access denied (java.net.SocketPermission 127.0.0.1:8086 connect,resolve)
java.security.AccessControlException: access denied (java.lang.reflect.ReflectPermission suppressAccessChecks)

Voor het genereren van de proxies aan de Java-kant, voor een .NET DLL is het van belang dit te doen met de instellingen van JDK 1.3.1, omdat Jinitiator 1.3.1.18 van Oracle Forms werkt met Java 1.3.1 en lager.
Dit kan worden gedaan in het java options scherm (jnbproxyGui, Project Java options) voor het builden van de proxy JAR.

Zorg ervoor dat de locatie van java.exe naar een Java 1.3 JVM verwijst om er zeker van te zijn dat de proxies tegen de juiste Java-versie voor Oracle Forms worden gegenereerd. In het verlengde hiervan moet de optie 'Generate j2se 5.0-targeted proxies' uit staan. Verdere informatie over het genereren van proxies is te vinden op de website van JNBridge.
Indien je toch tegen een hogere Java-versie de proxies genereert voor Oracle Forms krijg je de volgende exeption:

exception, java.lang.UnsupportedClassVersionError: System/EventHandler
(Unsupported major.minor version 49.0)

In de formsweb configuratie moet een aantal jar-files worden toegevoegd aan de archive_jini parameter.

```
archive_jini=f90all_jinit.jar,/forms90/hsd65java/hst65.jar,testNetJava.
jar,proxies.

jar,jnbcore.jar,bcel-5.1-jnbridge.jar
```

Dit zijn, buiten de standaard jar files van forms, de volgende jar files:

- De Java applicatie met daarin de JavaDotNet klasse voor Forms; testNetJava.jar;
- De gegenereerde proxies, dit wordt gebruikt door de JavaDotNet klasse; proxies.jar;
- En de benodigde jar files van het JNBridge framework; jnbcore.jar, bcel-5.1-jnbride.jar.

Conclusie

JNBridgePro is een erg krachtig product. In dit artikel is een aantal mogelijkheden uiteengezet en aan de hand van voorbeeldcode aangetoond dat het ook mogelijk is om bijvoorbeeld met behulp van Oracle Forms te interacteren met het .NET platform.
Door middel van JNBridgePro is het mogelijk over te stappen naar een ander platform en daarbij de bestaande software te koppelen. Dit biedt mogelijkheden om gefaseerd software te migreren naar het nieuwe platform en legacy applicaties intact te houden.

Referenties:

www.jnbridge.com

Harry van Oosten is Java EE consultant bij Yenlo.

Neem nu een abonnement op

Optimize

Hét onafhankelijke vakblad voor de Oracle professional

Databases, application servers, software development en andere Oracle-gerelateerde onderwerpen worden allemaal uitvoerig behandeld in Optimize. Het vakblad is een must voor alle Oracle professionals in Nederland. Optimize gaat diep in op technische en op marktontwikkelingen met betrekking tot Oracle (partners) en concurrenten.

Optimize staat boordevol **praktische en professionele informatie**, geschreven voor en door Oracle professionals. In het blad vindt u het laatste (technische) nieuws uit de markt, een update van productlanceringen, praktijkverhalen, interviews met professionals uit de Oracle wereld en veel technische artikelen van hoog niveau over met name dba-onderwerpen, software development en Oracle Applications. Optimize is ook het publicatieplatform van de Oracle Business Club, de Nederlandse vereniging van Oracle leveranciers. De O.B.C. levert iedere editie een redactionele bijdrage.

Optimize heeft een **uitgebreide website** met onder andere:

- actueel nieuws uit de Oracle-markt,
- een actuele agenda met relevante events,
- het online archief met alle artikelen uit het blad, te downloaden door abonnees,
- een overzicht van relevante vacatures voor Oracle specialisten.

U kunt zich kosteloos abonneren op de **e-mail nieuwsbrief** die eenmaal per 3 weken verschijnt en u volledig op de hoogte houdt van ontwikkelingen op Oracle-gebied. Naast de zes edities van Optimize en de uitgebreide website krijgt u als abonnee korting op de **seminars** die speciaal voor u worden georganiseerd.

Nog geen abonnee?
Meld u online aan op **www.optimize.nl**. Het eerste jaar profiteert u van bijna 50% korting voor nieuwe abonnees.

www.optimize.nl

OPTIMIZE

Array PUBLICATIONS