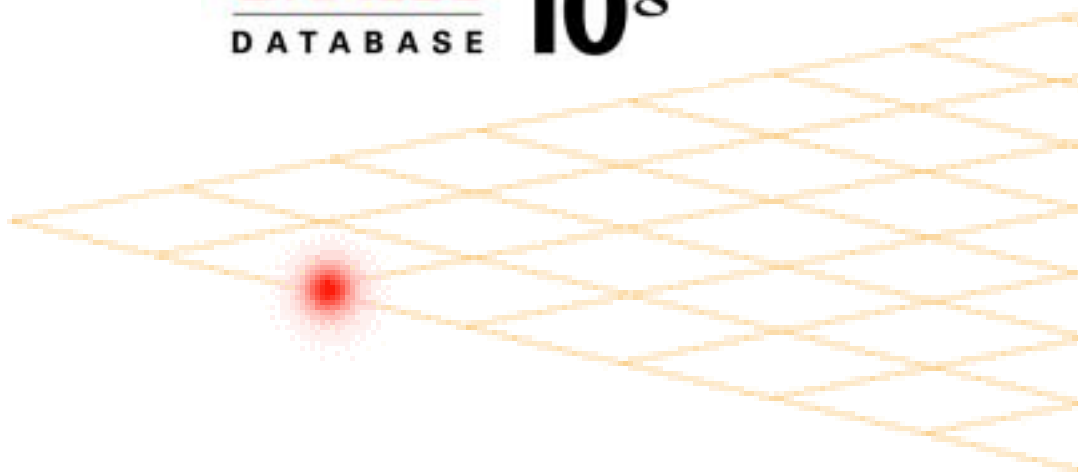


意外と簡単!?.NET で Oracle

- .NET 開発 with ODP.NET -

「0040 から ODP.NET 移行編」

ORACLE[®] 10^g
DATABASE



Creation Date:	Aug. 3, 2004
Last Update:	Sep 28, 2004
Version:	1.0



Document Control

Internal Use Only

Author

Hiroshi Ota

Change Logs

Date	Author	Version	Change Log
Aug. 3, 2004	Hiroshi Ota		Created.

Reviewers

Name	Position

Approvals

<Approver 1> _____

<Approver 2> _____

Distribution

Copy No.	Name	Location

はじめに

「意外と簡単!?.NET で Oracle」シリーズは、Microsoft Visual Studio.NET を使用して Oracle10g 対応アプリケーションをこれから開発されるかた向けに作成しております。.NET からオラクルへの接続にはさまざまな方法が存在しますが、「意外と簡単!?.NET で Oracle」シリーズではオラクル社が提供している Oracle Data Provider for .NET を利用しており、開発言語は Visual Basic.NET を使用しております。今回は「oo4o(Oracle Objects for OLE)から ODP.NET 移行編」ということで、今まで Visual Basic(4, 5, 6)と oo4o を利用してアプリケーションを開発されていたかたむけに、どのようにして既存のアプリケーションを.NET 対応するかを説明します。

「意外と簡単!?.NET で Oracle」シリーズが.NET 開発者でオラクルを利用したい方のシステム構築の一助になれば幸いです。

「意外と簡単!?.NET で Oracle」シリーズは以下の3つの構成を予定しています。

1. スマートクライアント編
2. Web アプリケーション編「ASP.NET」
3. oo4o(Oracle Objects for OLE)から ODP.NET 移行編（本書）

「意外と簡単!?.NET で Oracle」シリーズの「oo4o から ODP.NET 移行編」は、以下の7つの内容から構成しております。

1. ODP.NET にするメリット
2. oo4o と ODP.NET アーキテクチャの違い
3. oo4o と ODP.NET コネクションの違い
4. データアクセスの違い
5. PL/SQL 連携方法についての違い
6. トランザクション制御についての違い
7. ADO からの移行

「意外と簡単!?.NET で Oracle」シリーズにおける開発環境

- ・ データベース・サーバー
 - OS : Microsoft Windows 2000 Professional + SP4
 - RDBMS : Oracle Database 10g Standard Edition for Windows
- ・ アプリケーション・サーバー
 - OS : Microsoft Windows 2003 Enterprise
 - AP サーバー : Microsoft Internet Information Services 6.0
- ・ 開発クライアント
 - OS : Microsoft Windows 2000 Professional + SP4
 - 開発ツール : Microsoft Visual Studio .NET 2003
Microsoft Visual Studio 6.0

システム構成図



ODP.NET にするメリット

Oracle Data Provider for .NET (ODP.NET) は Microsoft .NET 環境 (以下、.NET) から Oracle データベースへの最適なデータ接続を提供するミドルウェアです。OLE DB .NET や ODBC .NET と違い、ODP.NET はデータアクセスのブリッジを経由しないネイティブ設計なので、最も効率的かつ高機能なデータベース接続を提供します。また、ODP.NET は Oracle に特化しているデータプロバイダであり、以下の固有の機能を実装しています。

- PL/SQL の完全なサポート
- ネイティブな Oracle データ型のサポート
- LOB 型、REF CURSOR、DATE 型など
- 接続プーリング
- 配列バインド
- グローバル化
- Unicode の完全サポート
- トランザクション
- Microsoft Transaction Server との連携
- XML DB のサポート
- 透過的アプリケーション・フェイルオーバー (TAF)

また、ODP.NET は .NET Framework が提供している API とのシームレスな連携が可能ですので、.NET 環境では Oracle Objects for OLE(以下、oo4o)に比べ開発生産性が向上します。また、.NET 環境から oo4o を使用した場合、Runtime Callable Wrapper(以下、RCW)が生成されますので、その分処理が遅くなります。RCW とは、マネージドコードが COM サーバーを呼び出す場合のアンマネージとマネージの相違をラッピングする機能になります。

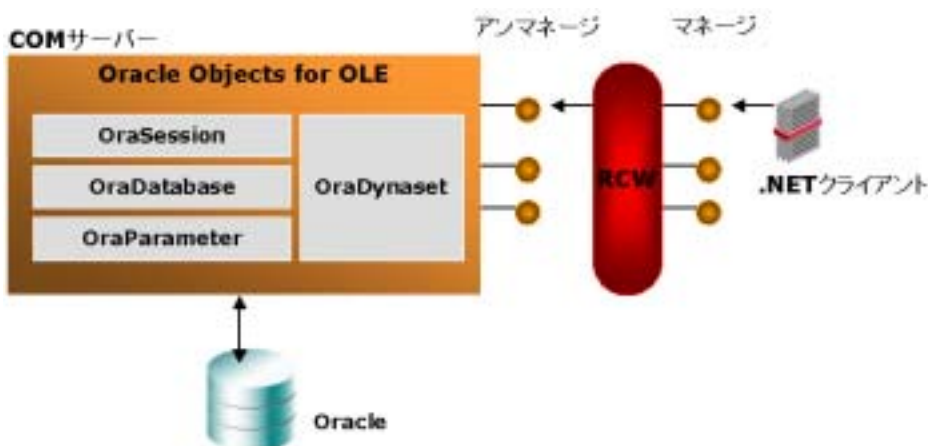


図1 RCW について

ODP.NET と oo4o の違いをまとめると、以下のようになります。

ODP.NET	oo4o
Oracle ネイティブなドライバー	COM サーバーである
データ・アクセスにブリッジが入らない	RCW によるデータ変換等のオーバーヘッドが発生する
Oracle 固有の機能のサポート	Oracle 固有の機能のサポート

表 1 ODP.NET と oo4o の違い

メモ：.NET 環境から oo4o を使用する場合は、Visual Basic.NET (以下 VB.NET) でのみ使用可能です。Visual C#.NET や Visual C++.NET では動作保証されていませんのでご注意ください。

oo4o と ODP.NET アーキテクチャの違い

oo4o と ODP.NET では Oracle へのデータアクセス方法が大きく異なります。その中でも最も大きな違いは、ADO.NET (ODP.NET は ADO.NET に準拠しています) から採用された、非接続型でのデータアクセス方式です。非接続型の特長としては、DataAdapter 経由でメモリ内の DataSet を利用します。したがって、Oracle データベースとの接続が切れている状態でもデータにアクセスが可能です。

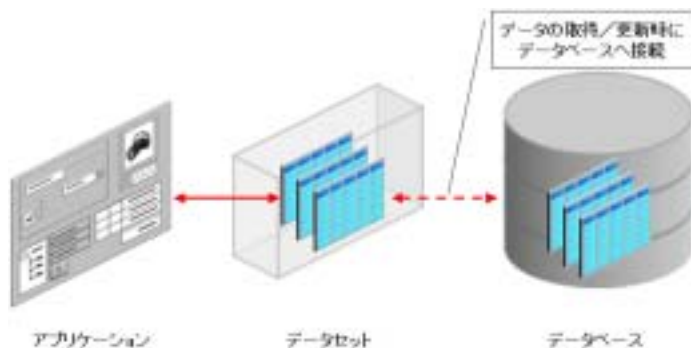


図 2 非接続型でのデータアクセス

逆に、接続型とは Command オブジェクトを利用し、SQL コマンドを直接発行するようになります。したがって、処理が終了するまで Oracle との接続を維持しなければなりません。oo4o には非接続型でのデータアクセスの概念がありませんので、oo4o から ODP.NET へデータプロバイダを変更する際には、新たにこの非接続型でのデータアクセス手法を理解する必要があります。非接続型でのデータアクセス手法の詳細は、「3. データアクセスの違い」で説明します。

oo4o と ODP.NET コネクションの違い

アプリケーションが Oracle データベースを使用して SQL を実行するためには、Oracle データベースに

SQL 文を渡す手段が必要です。このとき、アプリケーションと Oracle データベースを結ぶ SQL 実行の通信路が必要になります。この通信路のことを「コネクション」と言います。oo4o と ODP.NET のコネクションの大きな違いは、以下の 2 つが挙げられます。

- コネクションプーリングの動作
- 非接続型でのデータベースへの自動接続 / 自動切断

従来の Visual Basic によるクライアント / サーバーアプリケーション(以下、C/S)では、クライアント毎にコネクションが確立されていました。クライアント数が少ない場合には問題ありませんが、多い場合には、コネクション数が増大し、結果として、データベースサーバーに負荷がかかってしまいます。.NET アプリケーションでは、ASP.NET や XML WEB サービスを利用したスマートクライアントシステムのような、Windows フォームを利用する場合でも、Web サービス経由のスマートクライアントを利用することができ、コネクションプーリングが利用できます。その場合、中間層のアプリケーションサーバーがデータベースへのコネクションをプーリングし、複数のクライアントがプーリングされたコネクションを利用することにより、アプリケーションがコネクションを確立するためにかかる負荷を軽減させることが可能です。そのような機能を「コネクションプール」と言います。しかし、C/S システムではクライアント毎にコネクションが確立されるため、コネクションプールを使用したコネクションの一元化が有効になりませんのでご注意ください。

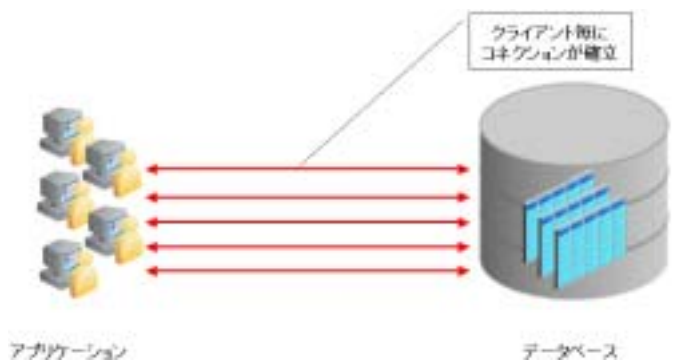


図3 C/S アプリケーションでのデータアクセス

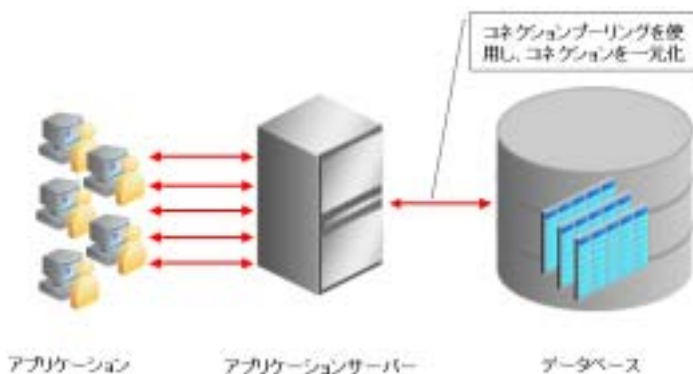


図4 WEB アプリケーションでのデータアクセス

ADO.NET でのデータベースへの接続はコネクションプールがデフォルトで有効になっており、ADO.NET に準拠した ODP.NET も同様にコネクションプールはデフォルトで有効になっております。oo4o でコネクションプールを利用する場合には、明示的にコネクションプールを有効にするためのコーディングが必要になります。具体的には、OraSession オブジェクトの CreateDatabasePool メソッドをコールします。

```
'OraSessionClass をオブジェクト化して、OracleInProcServer との接続を確立する。
Set OraSession = CreateObject("OracleInProcServer.XOraSession")

'Database との接続を管理する OraDatabase オブジェクト
OraSession.CreateDatabasePool 2, 40, 200, "ORCL", "SCOTT/TIGER", 0

'サービス名、ユーザーID、パスワードを指定して OraDatabase オブジェクトを生成
Set OraDatabase = OraSession.GetDatabaseFromPool(100)
```

リスト.1 oo4o でのコネクションプールの利用

ODP.NET では、以下のコードのように、Oracle データベースへの接続に最低限必要な接続文字列情報、User ID/Password/Data Source の指定のみで、コネクションプールが有効になります。

```
Imports Oracle.DataAccess.Client
Imports Oracle.DataAccess.Types

Public Class fmMainMenu
    Inherits System.Windows.Forms.Form
    (略)
    Private Sub DbConnect()
        Dim conn As New OracleConnection
        conn.ConnectionString = "Data Source=orcl;User id=scott;Password=tiger"
        conn.Open()
    End Sub
```

リスト.2 ODP.NET でのコネクションプールの利用(デフォルトで有効)

コネクションプールで使用する接続文字列属性とデフォルト値は以下のようになっております。

接続文字列属性	デフォルト値	説明
Pooling	TRUE	接続プーリングを有効または無効にします。
Connection Lifetime	0	接続の最長存続期間(秒)。
Connection Timeout	15	プールから空いた接続を取得するまで待機する最長時間(秒)。
Max Pool Size	100	プール内の最大接続数。
Min Pool Size	1	プール内の最小接続数。
Incr Pool Size	5	プール内のすべての接続が使用されている場合に確立される接続の数を制御します。
Incr Pool Size	5	プール内のすべての接続が使用されている場合に確立される接続の数を制御します。
Decr Pool Size	1	確立されているが使用されていない接続の数が多すぎる場合にクローズされる接続の数を制御します。
Validate Connection	FALSE	プールから発生した接続の検証の有効化または無効化。

表2 コネクションプールに関する接続文字列属性

ODP.NET でコネクションプールを明示的に有効にするには、OracleConnection オブジェクトの ConnectionString プロパティの接続文字列属性で設定します。

```
Dim conn As OracleConnection = New OracleConnection
'コネクションプールを明示的に設定
conn.ConnectionString = "User Id=SCOTT;Password=TIGER; Data Source=ORCL;" & _
    "Pooling=True;Min Pool Size=2;Max Pool Size=50"
conn.Open()
```

リスト.3 ODP.NET でのコネクションプールの利用(デフォルトで有効)

コネクションプールは、接続プーリング・サービスによって、プールを一意に識別するためのシグネチャとして ConnectionString を使用して作成されます。シグネチャとして ConnectionString は大文字・小文字も区別します。例えば、以下のような接続文字列ですと、Data Source の指定が大文字と小文字で違うので、それぞれに別々のコネクションプールが作成されてしまいます。

```
"User Id=scott;Password=tiger;Data Source=orcl"
"User Id=scott;Password=tiger;Data Source=ORCL"
```

リスト.4 コネクションプールの生成単位

メモ：ConnectionString プロパティでサポートされる接続文字列属性の一覧は、OTN の「[Oracle® Data Provider for .NET 開発者ガイド](#)」にて詳細に説明してありますので、そちらをご覧ください。

コネクションプール以外に、Oracle への接続に関して、ODP.NET は oo4o にはない以下の固有の機能があります。

tnsname.ora を使用しない接続

接続文字列属性の Oracle への接続文字列には、「Data Source」として、データベースの場所とデータベース・サービスの名前を示す接続記述子を指定します。具体的には tnsnames.ora ファイルに記述されているネット・サービス名です。

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = orcl)
    )
  )
```

リスト.5 TNSNAMES.ORA のネットサービス名の例

tnsnames.ora ファイルは、クライアント毎に配置する必要があり（Web アプリケーションの場合は、IIS がクライアントとなる）、多数のクライアントが存在する場合、そのファイルを配布するのも大変ですし、そのファイルに変更が必要になった場合は、さらに大変です。ODP.NET では、通常であれば、tnsnames.ora ファイルに記述する内容をアプリケーション・コード内に記述することが可能です。

```
Dim conn As New OracleConnection
Dim sb As New System.Text.StringBuilder
```

```

sb.Append("User Id=scott; Password=tiger;")
sb.Append("Data Source=(DESCRIPTION = (ADDRESS_LIST = ")
sb.Append("(ADDRESS = (PROTOCOL = TCP)(HOST = localhost)")
sb.Append("(PORT = 1521)))(CONNECT_DATA = (SERVER = DEDICATED)")
sb.Append("(SERVICE_NAME = ORCL));")
conn.ConnectionString = sb.ToString
conn.Open()
conn.Close()

```

リスト.6 tnsname.ora を使用しない接続

オペレーティング・システム認証と特権接続

ConnectionString 属性の User Id を” / “に設定することにより、データベース・ユーザーの認証に Windows ユーザー・ログイン資格証明を使用できます。また、DBA Privilege 属性を介して SYSDBA 権限または SYSOPER 権限のいずれかを使用して Oracle データベースに接続できます。

```

Dim cnn As New OracleConnection

認証に Windows ユーザー・ログイン資格証明を使用し、DBA Privilege 属性に「SYSDBA」を指定
cnn.ConnectionString = "User Id=/;Data Source=orcl;DBA Privilege=SYSDBA"
cnn.Open()

MsgBox("Connect OK!!")
cnn.Close()

```

リスト.7 オペレーティング・システム認証と特権接続

パスワードの期限切れ

Oracle ユーザーのパスワードが期限切れだった場合、新しいパスワードで接続をオープンすることが可能です。

```

Dim cnn As New OracleConnection
cnn.ConnectionString = _
"User Id=scott;Password=tiger;Data Source=ora10g"
Try
    cnn.Open()
Catch

```

```
cnn.OpenWithNewPassword("panther")
End Try
```

リスト.8 パスワードの期限切れ

データアクセスの違い

ODP.NET は、ADO.NET に準拠したデータプロバイダであり、ADO.NET と同じように「接続型データアクセス」と「非接続型データアクセス」を利用して Oracle データベースのデータを取得することが可能です。oo4o では「非接続型データアクセス」という概念がなく、どちらかという「接続型データアクセス」に近いアクセスモデルになります。oo4o と ODP.NET で使用するオブジェクトを比較すると、以下のようになります。

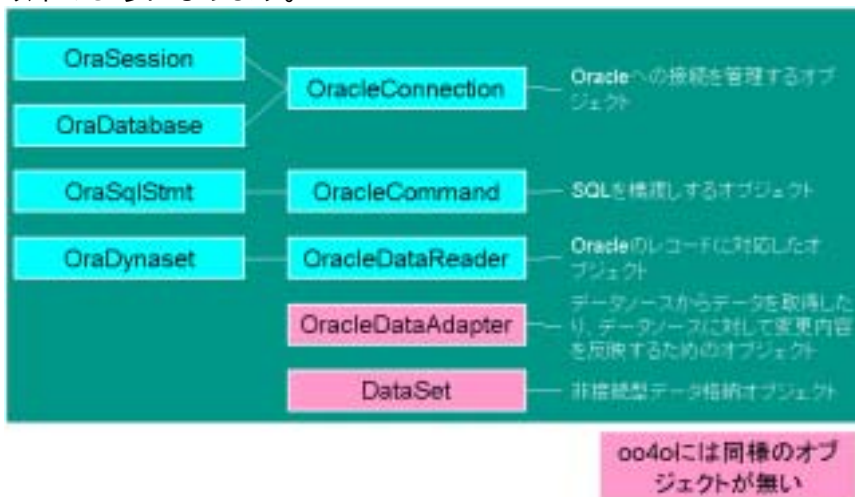


図5 oo4o と ODP.NET で使用するオブジェクトの比較

oo4o から ODP.NET へ移行した際に、「接続型データアクセス」と「非接続型データアクセス」でコードにどのような違いがあるのかを説明します。

接続型データアクセスでの比較

oo4o と ODP.NET の「接続型データアクセス」で使用するオブジェクトの比較は以下のようになります。

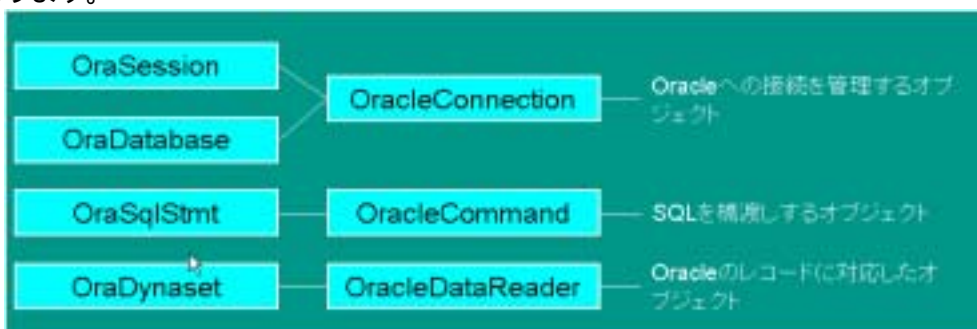


図6 接続型での oo4o と ODP.NET で使用するオブジェクトの比較

上記のオブジェクトを使用したコードをそれぞれ比較してみましょう。oo4o から emp 表の値を取得し、デバックウィンドウに結果を表示するコードは以下のようになります。

```
Dim OraDynaset As OraDynaset

'OraDynaset オブジェクトを生成
Set OraDynaset = _
    OraDatabase.CreateDynaset("SELECT empno,ename FROM emp", ORADYN_READONLY)

'デバックウィンドウに値を表示
Do Until OraDynaset.EOF
    Debug.Print OraDynaset.Fields("empno").Value + " / " + _
        OraDynaset.Fields("ename").Value

    OraDynaset.MoveNext
Loop
```

リスト.9 oo4o でのデータの取得サンプル

上記のコードを ODP.NET を使用したコードにすると以下のようになります。

```
Dim conn As New OracleConnection
conn.ConnectionString = _
    "User ID=scott;Password=tiger;Data Source=orcl"
conn.Open()
Dim cmd As New OracleCommand

'CommandText に SQL を設定する。
cmd.Connection = conn
cmd.CommandText = "select empno, ename from emp"

'OracleCommand オブジェクトの ExecuteReader メソッドを実行して、OracleDataReader オブジェクトを生成
Dim rdr As OracleDataReader
rdr = cmd.ExecuteReader
```

```

'デバックウィンドウに値を表示
Do While rdr.Read
    Debug.WriteLine(CStr(rdr("empno")) + " / " + _
        rdr("ename"))
Loop

'接続を Close
rdr.Close()
cmd.Close()

```

リスト.10 ODP.NET でのデータの取得サンプル(接続型)

以上のように、接続型でのデータアクセスに関しては、oo4o と ODP.NET は類似しております。一つ違いがあるとすれば、oo4o では、SQL を発行した結果セットを「OraDynaset」に直接格納しているのに対して、ODP.NET では「OracleCommand」を経由して SQL を発行し、結果セットを OracleDataReader で取得しています。次に、更新処理の違いを比較してみます。

```

Dim OraSession As Object
Dim OraDatabase As Object
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
Set OraDatabase = OraSession.OpenDatabase("orcl", "scott/tiger", &H2&)
'更新 SQL の発行
OraDatabase.ExecuteSQL "update emp set ename='scott' where empno=6"

```

リスト.11 oo4o でのデータの更新サンプル(接続型)

```

conn.Open()
Dim cmd As New OracleCommand
cmd.Connection = conn
'更新 SQL の発行
cmd.CommandText = "update emp set ename='Bob' where empno=6"
cmd.CommandType = CommandType.Text
cmd.ExecuteNonQuery()

```

リスト.12 ODP.NET でのデータの更新サンプル

上記のように、ODP.NET はデータの取得 / 更新時には必ず OracleCommand を使用します。後程説明する、非接続型データアクセス使用時にも OracleCommand の使用は必須になります。一般的にデータを更新する方法に関しては、SQL を直接発行する方法と、結果セットに対してフィー

ルド単位で値を設定し更新する方法の2つがあります。oo4o では OraDynaset 使用し、以下のコードのようにフィールド指定でのデータの更新が可能です。

```
Set OraDatabase = OraSession.OpenDatabase( _
    "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
sSql = "SELECT * FROM MstCustomer Where CustomerId=" + TextCustomerId.Text + ""
Set OraDynaset = _
    OraDatabase.CreateDynaset(sSql, ORADYN_DEFAULT)
OraDynaset.Edit
OraDynaset("CustomerID") = TextCustomerId.Text
OraDynaset("CustomerNAME") = TextCustomerName.Text
OraDynaset("EmployeeNAME") = TextEmployeeName.Text
OraDynaset("kana") = TextKana.Text
OraDynaset("job") = TextJob.Text
OraDynaset("postcode") = TextPostCode.Text
OraDynaset("prefectures") = ComboPrefectures.Text
OraDynaset("address") = TextAddress.Text
OraDynaset("tel") = TextTel.Text
OraDynaset("fax") = TextFax.Text
OraDynaset("note") = TextNote.Text
OraDynaset.Update
```

リスト.13 oo4o でのデータの更新サンプル(フィールド指定)

しかし、ODP.NET の接続型データアクセスで利用する OracleDataReader は読み取り専用であるため、上記のようなフィールドを指定した更新が出来ません。ODP.NET では、非接続型データアクセスを使用してフィールドを指定した更新が可能です。

非接続型データアクセスでの比較

「非接続型データアクセス」で使用する ODP.NET のオブジェクトは以下のようになります。

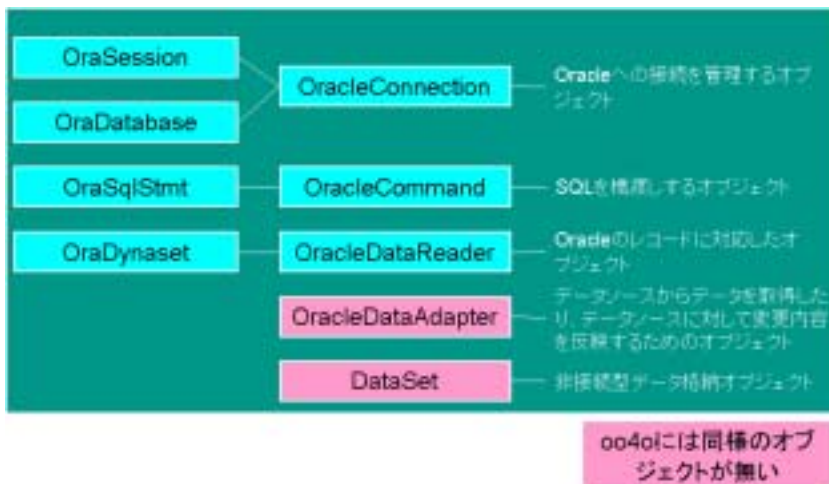


図6 非接続型での oo4o と ODP.NET で使用するオブジェクトの比較

「OracleDataAdapter」と「DataSet」オブジェクトは、非接続型データアクセスを行う場合の固有のオブジェクトになります。oo4o では非接続型データアクセスの概念がないので、「OracleDataAdapter」と「DataSet」同様のオブジェクトはありません。では、実際に ODP.NET を使用した「非接続型データアクセス」のコードを見てみましょう。

```

Dim cnn As New OracleConnection
Dim cmd As New OracleCommand
Dim dsList As New DataSet
Dim iCnt As Integer

cnn.ConnectionString = _
    "user id=scott;password=tiger;data source=orcl"
cmd.Connection = cnn

'DataAdapter を使用して、結果セットを DataSet に格納
cmd.CommandText = "Select * from emp"
Dim adp As New OracleDataAdapter(cmd)
adp.Fill(dsList, "EmpList")

'DataSet の内容をデバックウィンドウに表示
With dsList.Tables("EmpList")
    For iCnt = 0 To .Rows.Count - 1
        Debug.WriteLine(CStr(.Rows(iCnt).Item("empno")) + " / " + _
            .Rows(iCnt).Item("ename"))
    Next

```

```
End With
```

リスト.14 ODP.NET でのデータの取得サンプル(非接続型)

上記のコードは非接続型データアクセスでテーブルの情報を取得して、デバックウィンドウに表示するコードになります。DataSet に結果セットを格納するには、OracleDataAdapter の Fill メソッドを使用します。逆に、DataSet の内容を Oracle データベースに書き戻す作業時にも OracleDataAdapter の Update メソッドを使用します。



図7 非接続型でのデータの取得 / 更新

「リスト.14 ODP.NET でのデータの取得サンプル(非接続型)」のコードを見ると、接続のオープンとクローズをおこなっていないのがわかると思います。非接続型では、OracleDataAdapter の Fill メソッド、もしくは Update メソッドを呼び出したときに自動的に接続を確立し、処理が終了したら自動的に接続を切断します。DataSet へ値を格納後は、既に Oracle データベースとの接続が切れた状態になっています。このように、SQL を発行する瞬間にのみデータベースへ接続し、後の作業はデータベースへの接続が切断された状態でおこないます。このようなアクセス手法によることから、非接続型と呼ばれています。非接続型は、データベースへの接続時間を最小限にし、その結果、データベースサーバーの負荷を下げる事が可能になっています。DataSet に格納された値は、OracleDataAdapter の Update メソッドを使用し、Oracle データベースに更新された値を書き戻します。Update メソッドを実行すると、OracleDataAdapter の UpdateCommand, DeleteCommand, InsertCommand に設定された SQL が実行されます。



図8 OracleDataAdapter の Update メソッドをコールしたときの動作

OracleDataAdapter の UpdateCommand、DeleteCommand、InsertCommand それぞれの SQL は個別に設定することも可能ですが、OracleCommandBuilder を使用し、SQL を自動生成することが可能です。

```

Dim CustRow As DataRow
If dsCustomer.Tables("MstCustomer").Rows.Count = 0 Then
    CustRow = dsCustomer.Tables("MstCustomer").NewRow()
Else
    CustRow = dsCustomer.Tables("MstCustomer").Rows(0)
End If
CustRow.Item("CustomerId") = TextBoxID.Text
CustRow.Item("CustomerNAME") = TextBoxCompanyName.Text
    ~ 以下略 ~
If CustRow.RowState = DataRowState.Detached Then _
    dsCustomer.Tables("MstCustomer").Rows.Add(CustRow)
da.Update(dsCustomer.Tables("MstCustomer"))

```

リスト.15 ODP.NET でのデータの更新サンプル(非接続型)

上記のコードではカラム名を文字列で指定しており、データ型の復元もキャストにより行われているため、実行時エラーが発生する可能性があります。これを回避するためには、型付データセットというのを使用します。型付データセットとはテーブル名や列名などの定義情報を事前に取り込むことにより作成されるデータセットになります。型付データセットを利用すると、コードの可読性が向上しますし、インテリセンスによるコード補完も行われるため、コーディングミスも軽減されます。また、コンパイル時の名前チェックを有効化することも可能です。形なしデータセットと型付データセットのコードを比較すると、以下のようになります。

```

Dim strCustName As String = _
    dsCustomer.Tables("MstCustomer").Rows(0).Item("CustomerName")

```

リスト.16 形なしデータセットを使用したコード

```

Dim strCustName As String = dsCustomer.MstCustomer(0).CustomerName

```

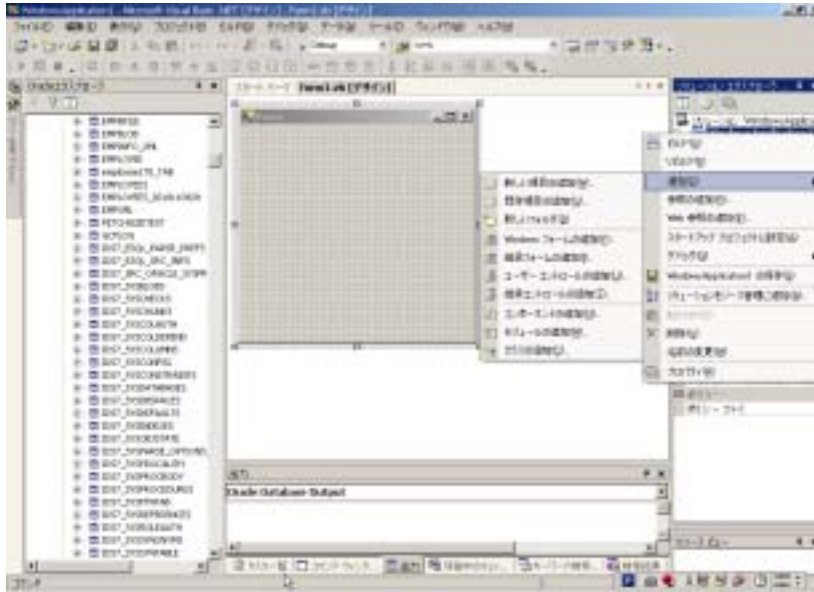
リスト.17 形付データセットを使用したコード

型付データセットの利用には、データセット定義ファイル(以下、xsd ファイル)を作成する必要があります。xsd ファイルは「Oracle Developer Tools for Visual Studio .NET」(以下、ODT)を使用すると簡単に作成できます。ODT のダウンロード/インストールについては、Oracle Technology Network(以下、OTN)の以下の「[.NET Developer Center](#)」に情報

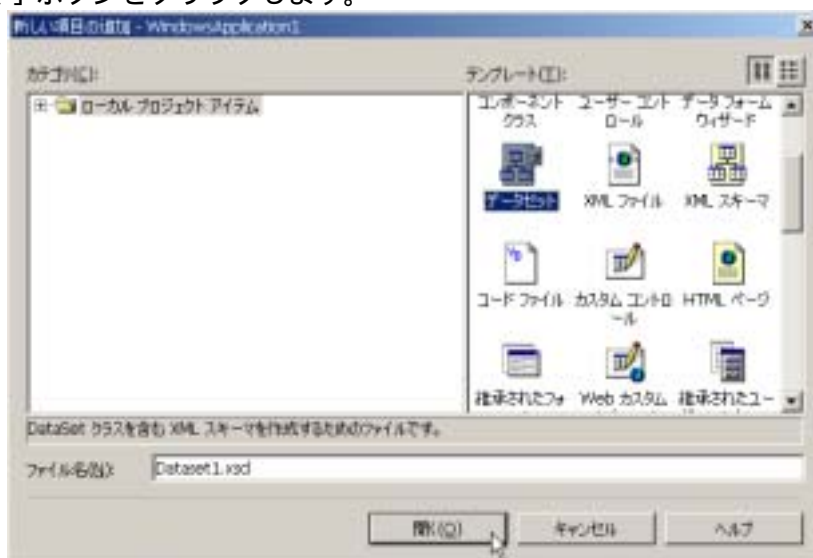
が掲載されておりますので、そちらをご覧ください。以下に ODT を使用して xsd ファイルを作成する方法について説明します。

1. xsd ファイルの新規作成

Visual Studio .NET から新規のプロジェクトファイルを作成し、プロジェクトファイルを右クリック -> 追加 -> 新しい項目の追加をクリックしてください。



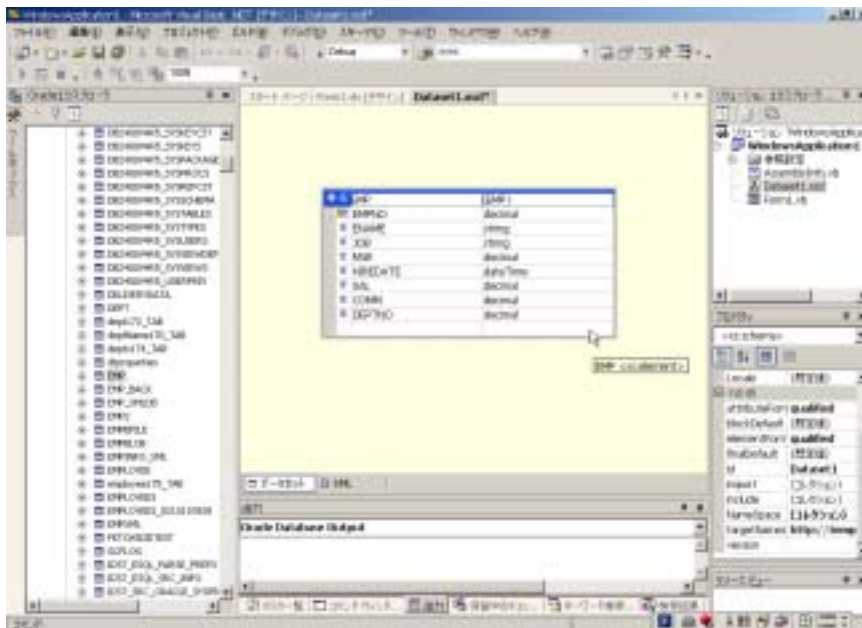
「新しい項目の追加」ウィンドウが表示されるので、「データセット」をクリックし、「開く」ボタンをクリックします。



2. xsd ファイルの作成

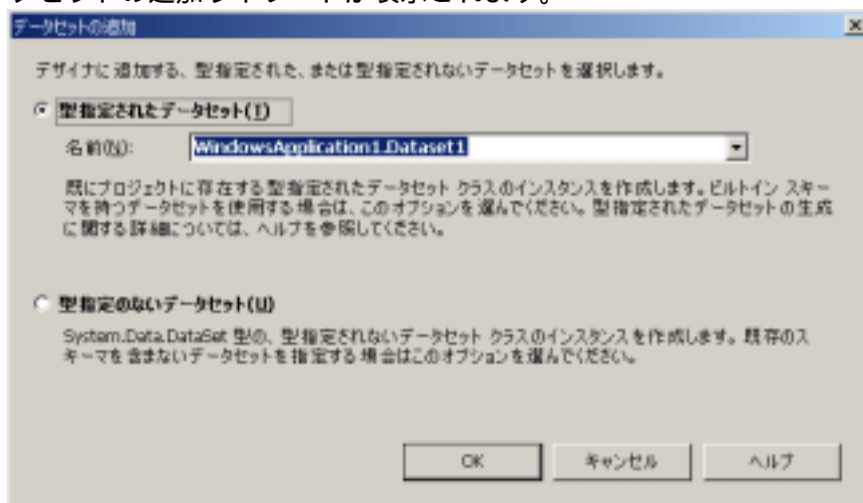
「Oracle エクスプローラ」から対象テーブルを選択し、新規に作成した xsd ファイルのデ

ザインウィンドウにドラッグ&ドロップすると、xsd ファイルが自動的に作成されます。



3. 型付データセットの作成

ツールボックスよりデータセットを選択し、フォームに貼り付けると、以下のようにデータセットの追加ウィザードが表示されます。



上記の画面で、「型指定されたデータセット(T)」を選択し、先ほど作成した xsd ファイルを選択後、「OK」ボタンをクリックすると、型付データセットが作成されます。

メモ：型付データセットの作成方法として、上記で説明した方法とは別の作成方法があります。「Oracle エクスプローラ」から対象テーブルをフォームに直接貼り付けると、OracleDataAdapter が自動生成されます。自動生成された OracleDataAdapter

を右クリックし、「プロパティ」ウィンドウから「DataSet を生成」をクリックすると、型付データセットが自動生成されます。

PL/SQL 連携方法についての違い

oo4o と ODP.NET での PL/SQL 連携の違いについて見てみましょう。まずは、以下の SQL を実行し、PL/SQL パッケージを作成します。

```
CREATE OR REPLACE PACKAGE SCOTT.pkg_ref AS
  CURSOR c1 IS SELECT ename FROM emp;
  TYPE empCur IS REF CURSOR RETURN c1%ROWTYPE;
  PROCEDURE GetEmpData(
    indeptno IN NUMBER,
    EmpCursor in out empCur );
END;

CREATE OR REPLACE PACKAGE BODY SCOTT.pkg_ref AS
  PROCEDURE GetEmpData(indeptno IN NUMBER,
    EmpCursor in out empCur ) IS
  BEGIN
    OPEN EmpCursor FOR
      SELECT ename FROM emp WHERE deptno=indeptno;
  END GetEmpData;
END pkg_ref;
```

リスト.18 サンプルパッケージの作成

上記の PL/SQL パッケージの GetEmpData ファンクションは、emp 表の deptno 列を引数で取得し、該当する結果セットを Ref Cursor で取得しております。次に oo4o から上記の PL/SQL パッケージの GetEmpData ファンクションをコールするコードは以下のようになります。

```
Dim sSql As String

Set OraDatabase = OraSession.OpenDatabase( _
  "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
' deptno をパラメータで設定
OraDatabase.Parameters.Add "DEPTNO", 10, ORAPARM_INPUT
OraDatabase.Parameters("DEPTNO").serverType = ORATYPE_NUMBER
'GetEmpData ファンクションをコール
sSql = "Begin pkg_ref.GetEmpData(:DEPTNO, :EmpCursor); end;"
'Ref Cursor の結果セットを取得
Set OraDynaset = OraDatabase.CreatePlsqlDynaset(sSql, "EmpCursor", 0&)
```

```

' 結果をデバックウィンドウに表示
Do While Not OraDynaset.EOF
    Debug.Print OraDynaset("ename")

    OraDynaset.MoveNext
Loop

```

リスト.19 oo4o からストアプロシージャをコールし、Ref Cursor を取得するコード

ODP.NET の接続型でのコードは以下のようになります。

```

Dim sSql As String

Dim conn As New OracleConnection("User Id=Scott;Password=Tiger;Data Source=orcl")
conn.Open()
'GetEmpData ファンクションをコールする SQL を OracleCommand に設定
Dim cmd As New OracleCommand("pkg_ref.GetEmpData", conn)
cmd.CommandType = CommandType.StoredProcedure
' deptno をパラメータで設定
cmd.Parameters.Add("DEPTNO", 10)
cmd.Parameters.Add("empCursor", OracleDbType.RefCursor, ParameterDirection.Output)
'GetEmpData ファンクションをコール
cmd.ExecuteNonQuery()
'Ref Cursor の結果セットを取得
Dim dr1 As OracleDataReader = _
    CType(cmd.Parameters(1).Value, OracleRefCursor).GetDataReader
' 結果をデバックウィンドウに表示
Do While dr1.Read
    Debug.WriteLine(dr1("ename"))
Loop

```

リスト.20 ODP.NET からストアプロシージャをコールし、Ref Cursor を取得するコード(接続型)

Ref Cursor の結果セットの取得は、非接続型でのアクセスでも可能です。実際のコードは以下のようになります。

```

Dim conn As New OracleConnection("User Id=Scott;Password=Tiger;Data Source=orcl")
'GetEmpData ファンクションをコールする SQL を OracleCommand に設定
Dim cmd As New OracleCommand("pkg_ref.GetEmpData", conn)
cmd.CommandType = CommandType.StoredProcedure

```

```

'Ref Cursor の結果セットを取得
cmd.Parameters.Add("DEPTNO", 10)
cmd.Parameters.Add("empCursor", OracleDbType.RefCursor, ParameterDirection.Output)

' GetEmpData フังก์ションをコールし、結果を DataSet に格納
Dim dsData As New DataSet
Dim da As New OracleDataAdapter(cmd)
da.Fill(dsData, "data")

'Grid へ表示
DataGridEmp.SetDataBinding(dsData, "data")

```

リスト.21 ODP.NET からストアプロシージャをコールし、Ref Cursor を取得するコード(非接続型)

次に、複数の Ref Cursor を取得するコードを比較してみます。実行を確認するために、テスト用のパッケージを作成します。

```

CREATE OR REPLACE PACKAGE SCOTT.pkg_ref2 AS
  CURSOR c1 IS SELECT * FROM emp;
  CURSOR c2 IS SELECT * FROM dept;
  TYPE empCur IS REF CURSOR RETURN c1%ROWTYPE;
  TYPE deptCur IS REF CURSOR RETURN c2%ROWTYPE;
  PROCEDURE GetEmpDeptData(
    EmpCursor in out empCur,
    DeptCursor in out deptCur
  );
END;

CREATE OR REPLACE PACKAGE BODY SCOTT.pkg_ref2 AS
  PROCEDURE GetEmpDeptData(
    EmpCursor in out empCur,
    DeptCursor in out deptCur) IS
  BEGIN
    OPEN EmpCursor FOR SELECT * FROM emp;
    OPEN DeptCursor FOR SELECT * FROM dept;
  END GetEmpDeptData;
END pkg_ref2;

```

リスト.22 複数の Ref Cursor を返すパッケージの作成

上記の PL/SQL パッケージの GetEmpDeptData ファンクションは、emp 表と dept 表の 2 つの結果セットを返すコードになります。oo4o で複数の Ref Cursor の結果セットを取得するコードは以下のようになります。

```

Dim sSql As String

Dim OraSession As New OraSessionClass
Dim OraDatabase As OraDatabase

Set OraDatabase = OraSession.OpenDatabase( _
    "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
OraDatabase.Parameters.Add "EmpCursor", 0, ORAPARM_OUTPUT
OraDatabase.Parameters("EmpCursor").serverType = ORATYPE_CURSOR
OraDatabase.Parameters.Add "DeptCursor", 0, ORAPARM_OUTPUT
OraDatabase.Parameters("DeptCursor").serverType = ORATYPE_CURSOR
sSql = "Begin pkg_ref2.GetEmpDeptData(:EmpCursor,:DeptCursor); end;"
Set OraSqlStmt = OraDatabase.CreateSql(sSql, ORASQL_FAILEXEC)

Set EmpDynaset = OraDatabase.Parameters("EmpCursor").Value
Set DeptDynaset = OraDatabase.Parameters("DeptCursor").Value

Do While Not EmpDynaset.EOF
    Debug.Print EmpDynaset("ename")

    EmpDynaset.MoveNext
Loop

Do While Not DeptDynaset.EOF
    Debug.Print DeptDynaset("dname")

    DeptDynaset.MoveNext
Loop

OraDatabase.Parameters.Remove ("EmpCursor")
OraDatabase.Parameters.Remove ("DeptCursor")

```

リスト.23 oo4o から複数の Ref Cursor を取得するコード

同様に、ODP.NET の接続型で複数の Ref Cursor の結果を取得してみます。

```
Dim conn As New OracleConnection("User Id=Scott;Password=Tiger;Data Source=orcl")
Dim cmd As New OracleCommand("pkg_ref2.GetEmpDeptData", conn)
cmd.CommandType = CommandType.StoredProcedure

'REF CURSOR パラメータのバインド
cmd.Parameters.Add("EmpCursor", OracleDbType.RefCursor, ParameterDirection.Output)
cmd.Parameters.Add("DeptCursor", OracleDbType.RefCursor, ParameterDirection.Output)
cmd.ExecuteNonQuery()

'SQL 文の実行と Ref Cursor の使用
Dim dr1 As OracleDataReader = _
    CType(cmd.Parameters(0).Value, OracleRefCursor).GetDataReader
Dim dr2 As OracleDataReader = _
    CType(cmd.Parameters(1).Value, OracleRefCursor).GetDataReader

'RefCursor の内容をデバックウィンドウに表示
Do While dr1.Read
    Debug.WriteLine(dr1("ename"))
Loop
Do While dr2.Read
    Debug.WriteLine(dr2("dname"))
Loop
```

リスト.24 ODP から複数の Ref Cursor を取得するコード(接続型)

非接続型でも同様に複数の Ref Cursor を取得可能です。

```
Dim conn As New OracleConnection("User Id=Scott;Password=Tiger;Data Source=orcl")
Dim cmd As New OracleCommand("pkg_ref2.GetEmpDeptData", conn)
cmd.CommandType = CommandType.StoredProcedure

'REF CURSOR パラメータのバインド
cmd.Parameters.Add("EmpCursor", OracleDbType.RefCursor, ParameterDirection.Output)
cmd.Parameters.Add("DeptCursor", OracleDbType.RefCursor, ParameterDirection.Output)

'SQL 文の実行と Ref Cursor の使用
Dim dsData As New DataSet
```

```
Dim da As New OracleDataAdapter(cmd)
da.Fill(dsData, "data")

'Grid へ表示
DataGridEmp.SetDataBinding(dsData, "data")
DataGridDept.SetDataBinding(dsData, "data1")
```

リスト.25 ODP.NET から複数の Ref Cursor を取得するコード(非接続型)

トランザクション制御について

oo4o と ODP.NET のトランザクション制御の違いについて見ていきましょう。まずは、oo4o でのトランザクション制御のコードについて説明します。

```
'---データベースとの接続を開く
Dim OraSession As New OraSessionClass
Dim OraDatabase As OraDatabase
Set OraDatabase = OraSession.OpenDatabase( _
    "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
'トランザクションの開始
OraSession.BeginTrans
OraDatabase.ExecuteSQL "insert into emp(empno,ename) values(6,'Michel')
OraSession.CommitTrans
```

リスト.26 oo4o からのトランザクション制御

oo4o では、OraSession オブジェクトに対して、トランザクションの制御を行います。ODP.NET では、OracleConnection オブジェクトから、OracleConnection の BeginTransaction メソッドを実行し、ローカル・トランザクションを開始します。

```
'---データベースとの接続を開く
cnn.Open()
Dim cmd As New OracleCommand
cmd.Connection = cnn
'トランザクションの開始
Dim txn As OracleTransaction = cnn.BeginTransaction()
cmd.CommandText = "insert into emp(empno,ename) values(6,'Michel')
cmd.CommandType = CommandType.Text
cmd.ExecuteNonQuery()
txn.Commit()
```

リスト.27 ODP.NET からのトランザクション制御

また、ODP.NET では以下のように「SavePoint」を指定して、トランザクション内にセーブポイントを作成することが可能です。

```
Dim cnn As New OracleConnection

cnn.ConnectionString = "user id=scott;password=tiger;data source=orcl"
cnn.Open()
'トランザクションの開始
Dim txn As OracleTransaction = cnn.BeginTransaction()
Dim strSQL1 As String = "INSERT INTO emp (empno, ename) VALUES (1,'Employee1')"
Dim myCmd As New OracleCommand(strSQL1, cnn)
Dim res As Integer = myCmd.ExecuteNonQuery()
'SavePoint 'a' でトランザクションを保存
txn.Save("a")

Dim strSQL2 As String = "INSERT INTO emp (empno, ename) VALUES (2,'Employee2')"
Dim myCmd2 As New OracleCommand(strSQL2, cnn)
Dim res2 As Integer = myCmd2.ExecuteNonQuery()
'SavePoint 'b' でトランザクションを保存
txn.Save("b")

SavePoint 'a'までロールバックしコミット
txn.Rollback("a")
txn.Commit()
```

リスト.28 ODP.NET からのトランザクション制御(SavePoint の利用)

上記のコードでは、コード内でトランザクションの制御をおこなっています。その他に、COM+サービスを利用した自動トランザクション機能を利用し、トランザクションの制御をおこなうことも可能です。oo4o を利用している場合、データアクセス部分を COM コンポーネントとして作成し、コンポーネントサービスで COM コンポーネントの登録を行い、COM コンポーネント単位でトランザクションの管理をおこないます。

ADO からの移行

Visual Basic 6.0(以下、VB)から Oracle データベースへの接続ミドルウェアとして、oo4o 以外にも ADO を選択されている方も多いと思います。ここでは、VB で ADO を使い Oracle データベースにアクセスするアプリケーションを、どのように .NET 化すればよいか説明します。ADO と ODP.NET で使用するオブジェクトは以下のようになります。

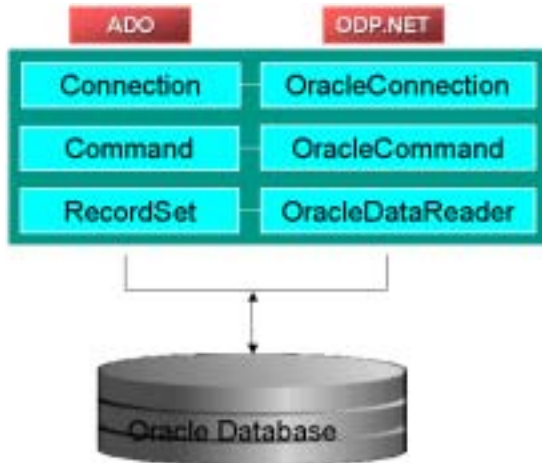


図11 ADOとODP.NETで使用するオブジェクトの比較

ADO では、以下のコードのように RecordSet を開いて、Movenext メソッドでシーケンシャルに読み込みながら処理を行うプログラミングスタイルが一般的だと思います。

```
Do Until RecordSet.Eof
  ~ データアクセス ~
  RecordSet.Movenext
Loop
```

リスト.29 ADO の RecordSet を利用したサンプルコード

以上のコードですと、ループ処理の間はデータベースと接続された状態になっています。ODP.NET の OracleDataReader は、ADO の RecordSet の ReadOnly , FowardOnly オプションで開いた状態に似ています。 .NET 環境からのデータアクセスは非接続型でのデータアクセスが主流になりますので、ADO から ODP.NET への移行も、oo4o から ODP.NET の移行と同様に、非接続型でのデータアクセス手法を習得する必要があります。OracleDataAdapter を経由した、DataSet へのデータの格納と、DataSet からデータベースへのデータの書き戻しの概念を学ぶ必要があります。

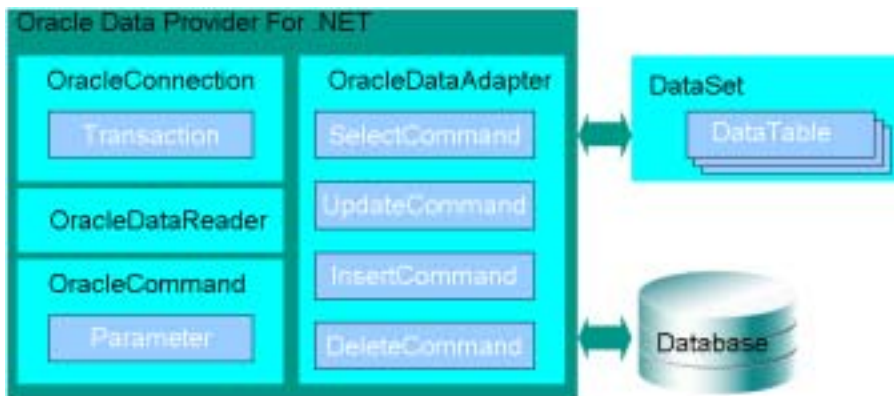


図12 非接続型でのデータアクセス

しかし、既存の ADO を利用して作成されたアプリケーションを ODP.NET に変更するには、かなりのコードの修正が必要となります。とりあえず、最も簡単に ADO を使用して作成されたアプリケーションを .NET 化する方法として、Visual Studio .NET に付属している、アップグレードウィザードを使用する方法があります。アップグレードウィザードとは、Visual Basic 6.0 プロジェクトを Visual Basic .NET プロジェクトにアップグレードするツールになります。コマンドラインから、Vbupgrade.exe の引数にプロジェクトを指定することでアップグレードできますし、Visual Basic 6.0 のプロジェクトを Visual Studio .NET で開くと、自動的にウィザードが起動し、プロジェクトをアップグレードすることもできます。

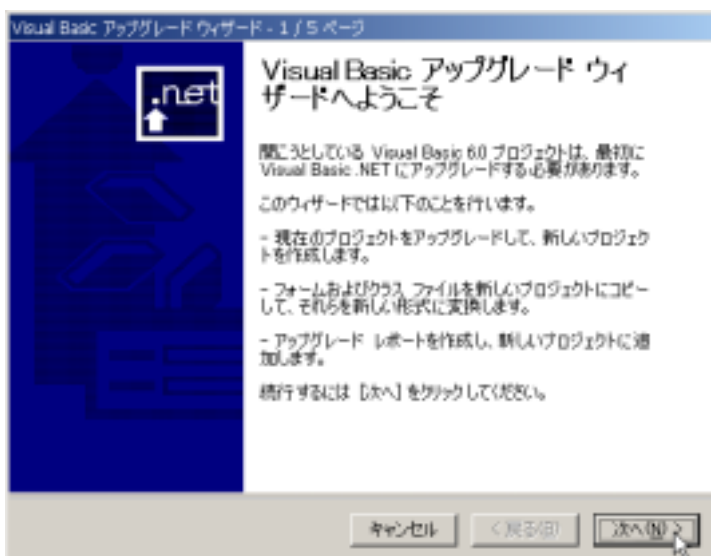


図13 アップグレード ウィザードの起動

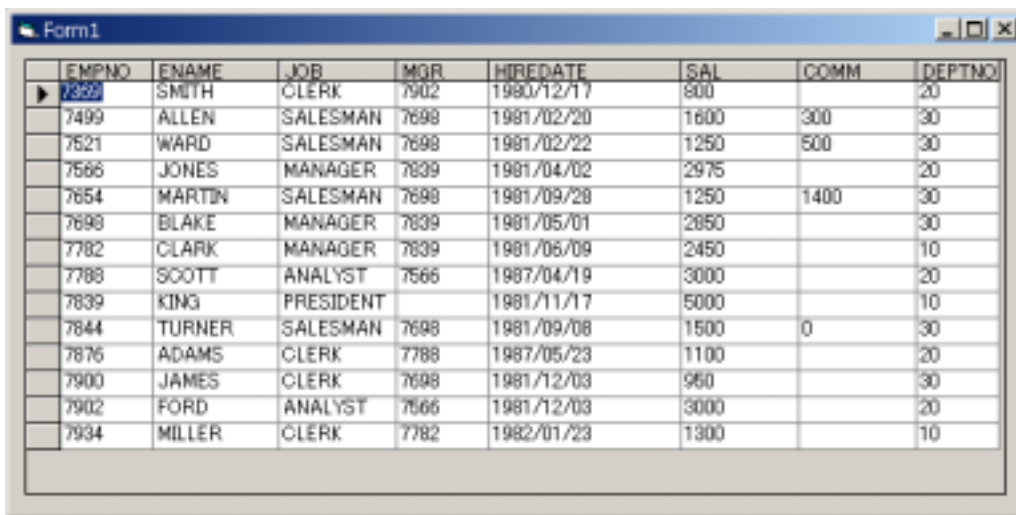
実際に、アップグレードウィザードを使用して Visual Basic 6.0 プロジェクトを Visual Basic .NET プロジェクトに変換してみましょう。まずは、Visual Basic 6.0 で作成された移行元のアプリケーションを作成します。Oracle データベースにアクセスし、Visual Basic 6.0 の DataGrid コントロールに emp 表

の内容を表示するアプリケーションのコードは以下のようになります。

```
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset

Set conn = New ADODB.Connection
conn.CursorLocation = adUseClient
conn.ConnectionString = _
    "Provider=OraOLEDB.Oracle.1;User ID=scott;Password=tiger;Data Source=orcl;"
conn.Open
Set rs = conn.Execute("SELECT * FROM emp", , adCmdText)
Set Me.DataGrid1.DataSource = rs
```

リスト.30 アップグレードウィザードを使用して、.NET 化したコード



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7320	SMITH	CLERK	7902	1980/12/17	800		20
7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
7566	JONES	MANAGER	7839	1981/04/02	2975		20
7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
7698	BLAKE	MANAGER	7839	1981/05/01	2850		30
7782	CLARK	MANAGER	7839	1981/06/09	2450		10
7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
7839	KING	PRESIDENT		1981/11/17	5000		10
7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
7876	ADAMS	CLERK	7788	1987/05/23	1100		20
7900	JAMES	CLERK	7698	1981/12/03	950		30
7902	FORD	ANALYST	7566	1981/12/03	3000		20
7934	MILLER	CLERK	7782	1982/01/23	1300		10

上記の VB で作成されたアプリケーションのプロジェクトファイルを Visual Studio .NET で開くと、アップグレードウィザードが起動し、プロジェクトを VB.NET へアップグレードできます。実際にアップグレードされたコードは以下のようになります。

```
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset

conn = New ADODB.Connection
conn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
conn.ConnectionString = _
    "Provider=OraOLEDB.Oracle.1;User ID=scott;Password=tiger;Data Source=orcl;"
```

```
conn.Open()

rs = conn.Execute("SELECT * FROM emp", , ADOB.CommandTypeEnum.adCmdText)

Me.DataGrid1.DataSource = rs
```

リスト.31 アップグレードウィザードを使用して、.NET 化したコード

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980/12/17	800		20
7499	ALLEN	SALESMAN	7698	1981/02/20	1600	300	30
7521	WARD	SALESMAN	7698	1981/02/22	1250	500	30
7566	JONES	MANAGER	7639	1981/04/02	2975		20
7654	MARTIN	SALESMAN	7698	1981/09/28	1250	1400	30
7698	BLAKE	MANAGER	7639	1981/05/01	2850		30
7782	CLARK	MANAGER	7639	1981/06/09	2450		10
7788	SCOTT	ANALYST	7566	1987/04/19	3000		20
7839	KING	PRESIDENT		1981/11/17	5000		10
7844	TURNER	SALESMAN	7698	1981/09/08	1500	0	30
7876	ADAMS	CLERK	7788	1987/05/23	1100		20
7900	JAMES	CLERK	7698	1981/12/03	950		30
7902	FORD	ANALYST	7566	1981/12/03	3000		20
7934	MILLER	CLERK	7782	1982/01/23	1300		10

アップグレードウィザードを使った場合、データベースへのアクセスは、ADO.NET に変換されるわけではなく、ADO のラッパークラスを使って ADO のまま動作するように変換されます。また、DataGrid コントロールも Visual Basic 6.0 のコントロールがそのまま使われています。確かに、この実装でも正しく動作します。しかも開発者は殆どコードを変更することなく、Visual Basic .NET に移行することができます。しかしながら、この実装方法には、いくつか注意しなければならない点があります。ラッパークラスを返して ADO を使うことにより、オーバーヘッドが発生し、従来の Visual Basic 6.0 で作成したアプリケーションよりパフォーマンスが低下します。本来、.NET Framework が提供する ADO.NET を使用することによる、パフォーマンスの向上や、非接続型のデータアクセスと言った .NET のメリットを十分に発揮できません。上記のコードを、アップグレードウィザードを使用せずに ODP.NET で実装したコードは以下のようになります。

```
Dim conn As New OracleConnection( _
    "User Id=Scott;Password=Tiger;Data Source=orcl")
Dim cmd As New OracleCommand("Select * from emp", conn)
Dim adp As New OracleDataAdapter(cmd)
Dim ds As New DataSet
adp.Fill(ds, "emplist")
DataGrid1.SetDataBinding(ds, "emplist")
```

リスト.32 ADO から ODP.NET へ手動でコードを変更したサンプル

コードの変更以外にも、フォームで使用しているコントロールの移行も必要になります。上記のアプリ

ケーションは、Visual Basic 6.0 で使用している DataGrid コントロールを Visual Studio .NET の DataGrid コントロールに変更しています。同じ DataGrid コントロールでも仕様は異なりますのでご注意ください。



日本オラクル株式会社

Copyright © 2005 Oracle Corporation Japan. All Rights Reserved.

無断転載を禁ず

この文書はあくまでも参考資料であり、掲載されている情報は予告なしに変更されることがあります。日本オラクル社は本書の内容に関していかなる保証もいたしません。また、本書の内容に関連したいかなる損害についても責任を負いかねます。

Oracle は米国 Oracle Corporation の登録商標です。文中に参照されている各製品名及びサービス名は米国 Oracle Corporation の商標または登録商標です。その他の製品名及びサービス名はそれぞれの所有者の商標または登録商標の可能性がります。