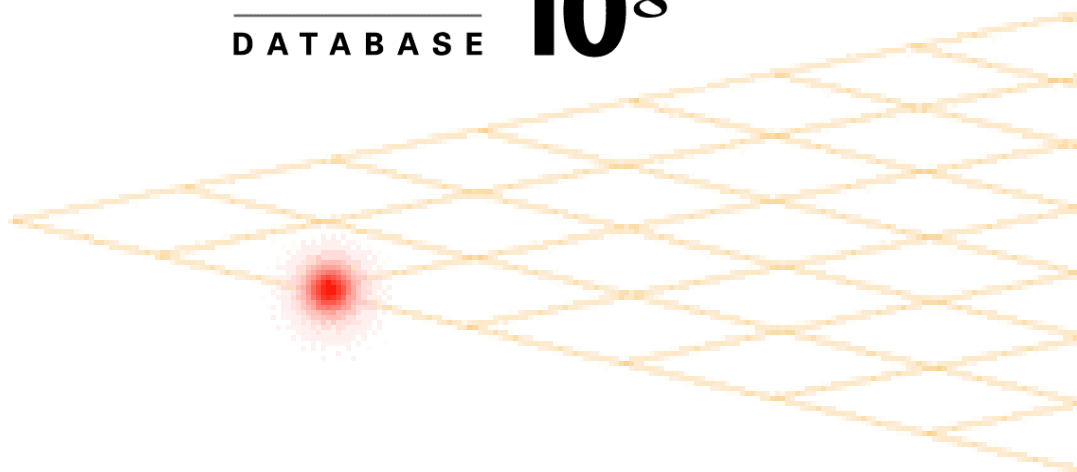


意外と簡単!?.NET で Oracle

- .NET 開発 with ODP.NET -

「ASP.NET 編」

ORACLE®
DATABASE 10^g



はじめに	2
ASP.NETでOracle Data Provider For .NETの使用	4
サンプルアプリケーションのインストール	6
ASP.NETからのLOBの扱い	11
ASP.NETアプリケーションのセキュリティについて	17
ODP.NETでのエラーハンドリング	20
ODP.NETとWebサーバーコントロールの連携	22
ODP.NETのデバッグ・トレース機能の利用	31
文キャッシングについて	32
グローバル化のサポート	34

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。実際のサンプルアプリケーションを提供することにより、単なるコーディング Tips にとどまらず、より実践的なアプリケーション開発の資料として構成するようにしております。

.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。今回のサンプルアプリケーションの説明はポイントとなる部分のみの説明になりますので予めご了承ください。

「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」(本書)
3. OO4O(Oracle Objects for OLE)から ODP.NET 移行編

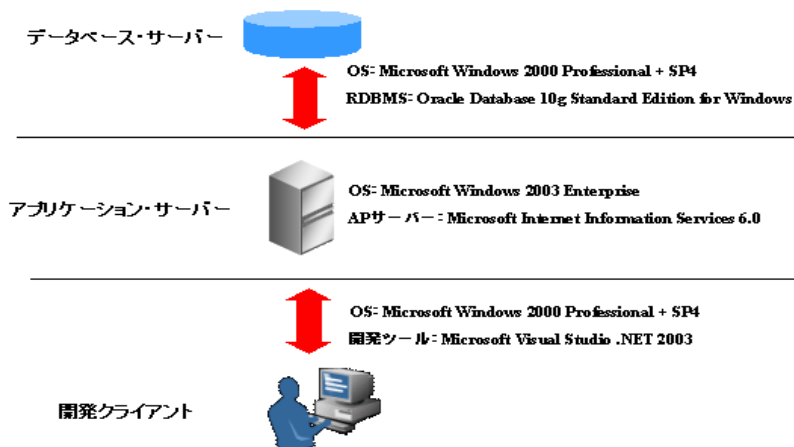
「意外と簡単!?.NET で Oracle」シリーズの「ASP.NET 編」は、以下の 9 つの内容から構成しております。

1. ASP.NET で Oracle Data Provider For .NET の使用
2. サンプルアプリケーションのインストール
3. ASP.NET からの LOB の扱い
4. PROXY 認証について
5. ODP.NET と Web サーバーコントロールの連携
6. ODP.NET のデバッグ・トレース機能の利用
7. OracleError クラスによるエラーハンドリング
8. 文キャッシングの利用
9. グローバリゼーションのサポート

「意外と簡単!?.NET で Oracle」シリーズにおける開発環境

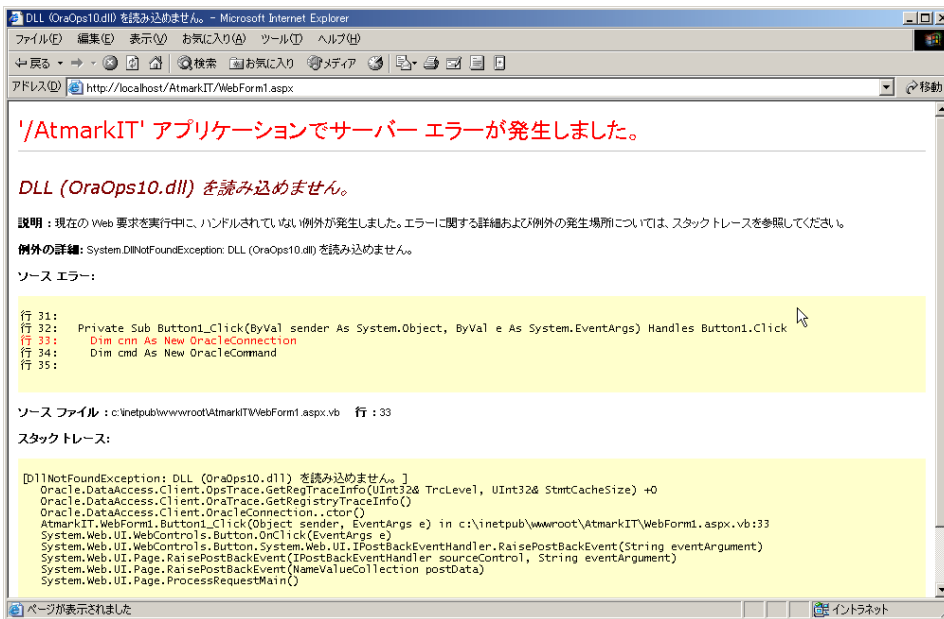
- ・ データベース・サーバー
 - OS : Microsoft Windows 2000 Professional + SP4
 - RDBMS : Oracle Database 10g Standard Edition for Windows
- ・ アプリケーション・サーバー
 - OS : Microsoft Windows 2003 Enterprise
 - AP サーバー : Microsoft Internet Information Services 6.0
- ・ 開発クライアント
 - OS : Microsoft Windows 2000 Professional +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

システム構成図



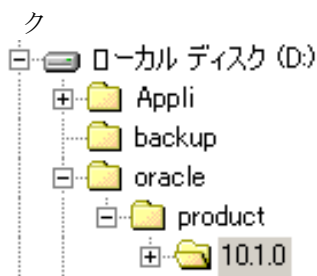
ASP.NET で Oracle Data Provider For .NET の使用

まず初めに、ASP.NET から Oracle Data Provider for .NET(以下 ODP.NET)を使用する際に必要な設定について説明します。IIS では ASP.NET スクリプトの実行時、ASPNET ユーザーとして実行されます。ASPNET ユーザーが ORACLE_HOME ディレクトリにアクセス権が無いと以下のようなエラーが発生しますのでアクセス権を設定する必要があります。

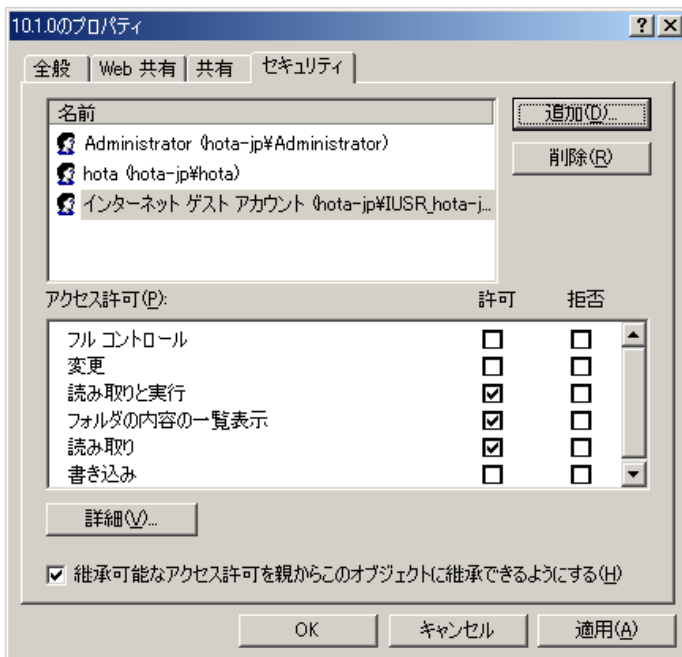


画面 1 ORACLE_HOME へのアクセス権が無いために発生するエラー画面

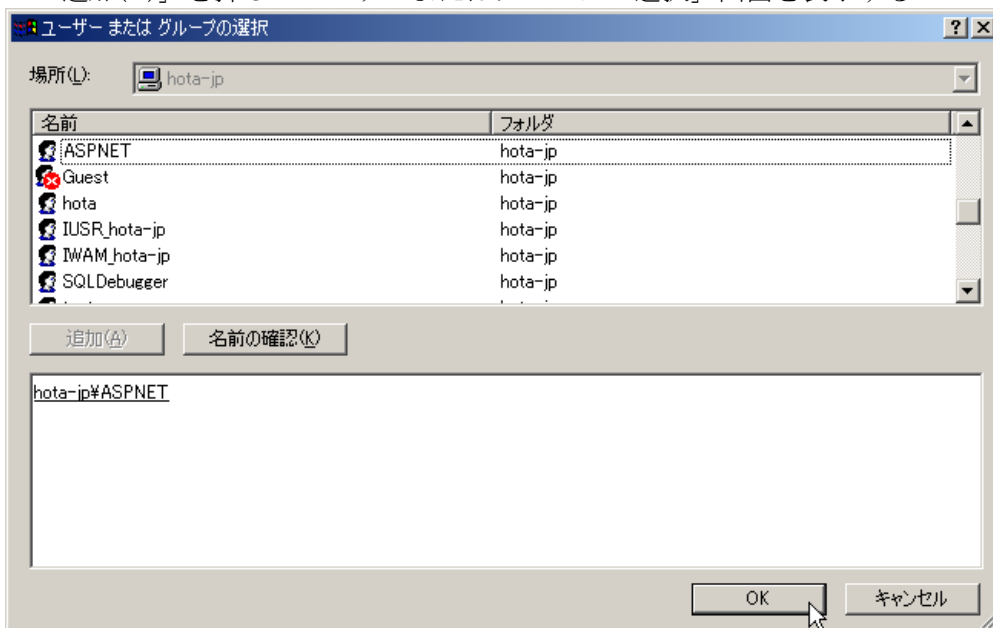
1. Administrator 権限のあるユーザーで Windows にログオンする
2. Windows の Explorer (Explorer.exe)を起動し、ORACLE_HOME ディレクトリを選択して右クリッ



3. 「プロパティ」を選択し「セキュリティ」タブを開く



4. 「追加(D)」 を押し「ユーザーまたはグループの選択」画面を表示する



5. 「場所」 からローカルマシンを選択し、上記の「ASPNET」を選択し「追加(A)」を押し「OK」を押して追加する。
6. 5.で追加したユーザーを選択し、アクセス許可(P)より「許可」の列にある「読み取りと実行」のチェックをつけ「適用(A)」に続いて「OK」を押す

以上の設定で IIS 環境から ODP.NET が使用可能になります。

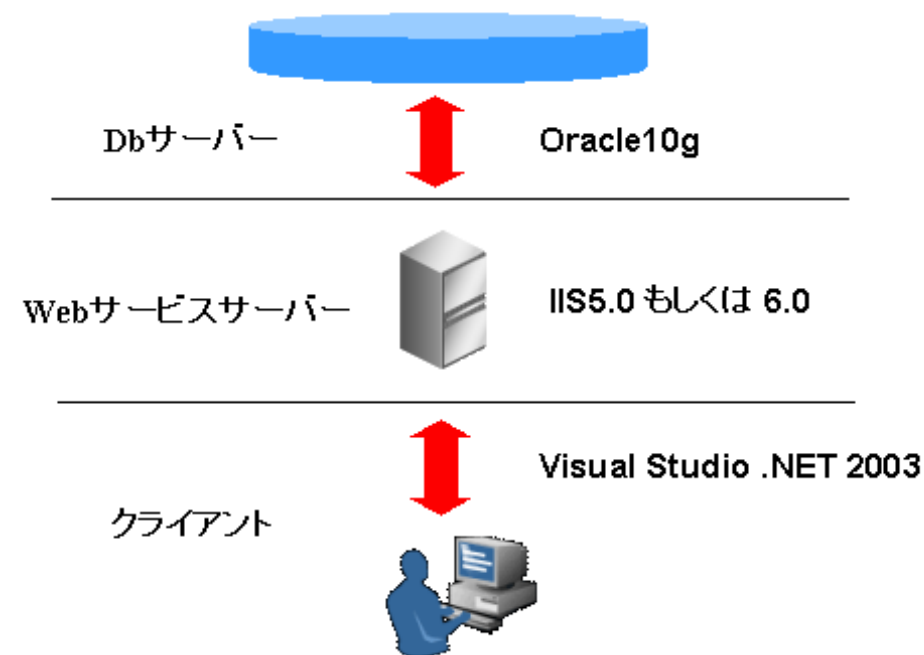
サンプルアプリケーションのインストール

サンプルアプリケーションは以下のサイトよりダウンロードできます。

<http://otn.oracle.co.jp/easy/dotnet/index.html>

ダウンロードファイル(sample.zip)を任意の一時フォルダに保存し、解凍してください。解凍しますと「data」、「StockManagement」、「WebServiceStockManagement」という 3 つのディレクトリが作成されます。今回のサンプルアプリケーションでは以下のように データベース・サーバー、Web アプリケーション・サーバー、クライアントの 3 つの構成になります。ダウンロードファイルを解凍後にそれぞれの端末で以下の作業を行ってください。

システム構成図



メモ： 今回のサンプルを動作させるためにOracle Database 10gをインストールし、データベースを構築しておく必要があります。Oracle Database 10gのインストール、初期データベースの作成などについては、OTN上の「意外と簡単!? Oracle Database 10g」をご参照下さい。 <http://otn.oracle.co.jp/beginner/oracle10g/index.html>

サンプルデータ・パッケージの作成 (データベース・サーバーでの設定)

今回のシステムで使用しているサンプルデータとパッケージを Oracle がインストールされたデー

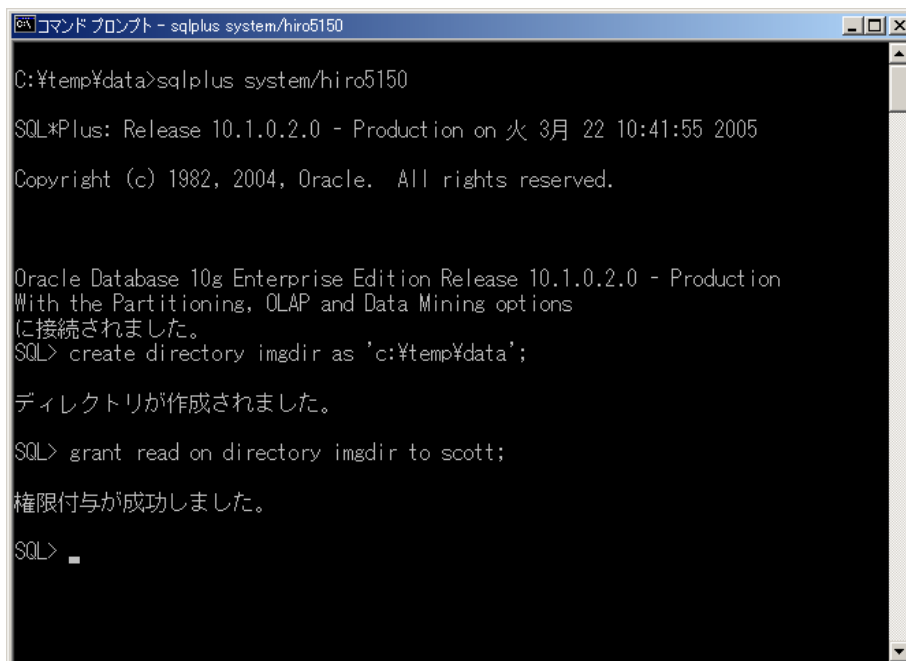
データベース・サーバー上にて、以下の手順で作成します。

1. ディレクトリの設定

今回のサンプルアプリケーションでは商品マスタ上の商品イメージを画像ファイルとして管理しております。サンプルデータ作成時にビットマップファイルを取り込みます。そのためにビットマップファイルが保管されているディレクトリを読み込み可能に設定する必要があります。ダウンロードファイル(sample.zip)を解凍すると作成される data ディレクトリを Oracle Database 10g がインストールされているサーバー上の任意のディレクトリにコピーして以下のコマンドを SQL*Plus 上で実行してください。

(data ディレクトリを c:¥temp¥data にコピーし、ユーザーscott に対してディレクトリを読み込み可能にする設定のサンプル)

- sqlplus system/<password>(SQL*Plus に system ユーザーでログイン)
- create directory imgdir as 'c:¥temp¥data';
- grant read on directory imgdir to scott;



```
コマンド プロンプト - sqlplus system/hiro5150
C:¥temp¥data>sqlplus system/hiro5150
SQL*Plus: Release 10.1.0.2.0 - Production on 火 3月 22 10:41:55 2005
Copyright (c) 1982, 2004, Oracle. All rights reserved.

Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
に接続されました。
SQL> create directory imgdir as 'c:¥temp¥data';
ディレクトリが作成されました。
SQL> grant read on directory imgdir to scott;
権限付与が成功しました。
SQL> _
```

画面 2 SQLPLUS で scott ユーザーに対してディレクトリ設定を行った画面

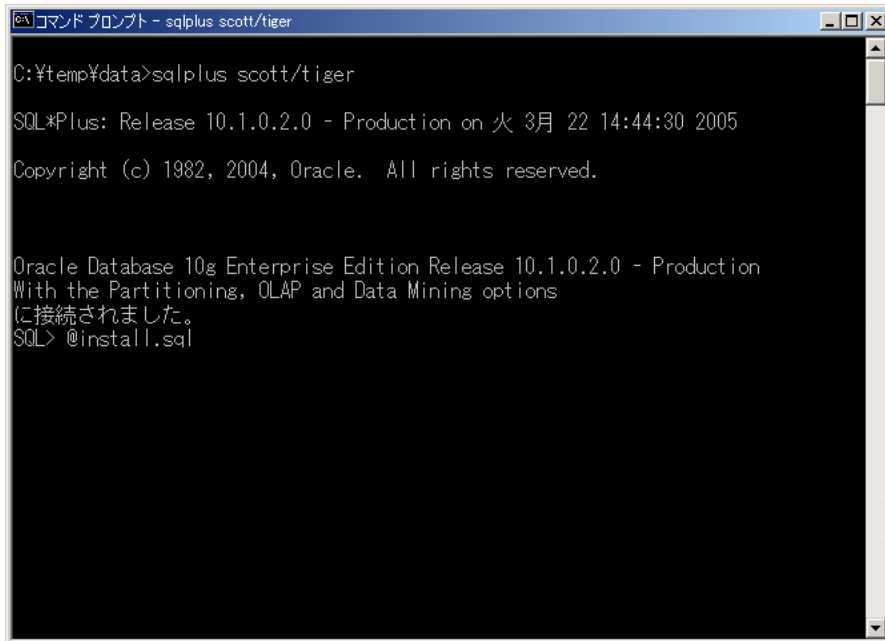
メモ： create directory により、実際にディレクトリが作成されるわけではなく、Oracle で管理するためのディレクトリオブジェクトを作成しております。

2. テーブルとサンプルデータのインストール

data ディレクトリの install.sql を SQL*Plus で実行しますと、テーブルとパッケージが作成され

まず、以下のように install.sql を実行してください。

- sqlplus scott/tiger (SQL*Plus にログイン)
- @install.sql (テーブルとパッケージの作成スクリプト SQL の実行)
- exit (SQL*Plus の終了)



```
コマンドプロンプト - sqlplus scott/tiger
C:\temp\data>sqlplus scott/tiger
SQL*Plus: Release 10.1.0.2.0 - Production on 火 3月 22 14:44:30 2005
Copyright (c) 1982, 2004, Oracle. All rights reserved.

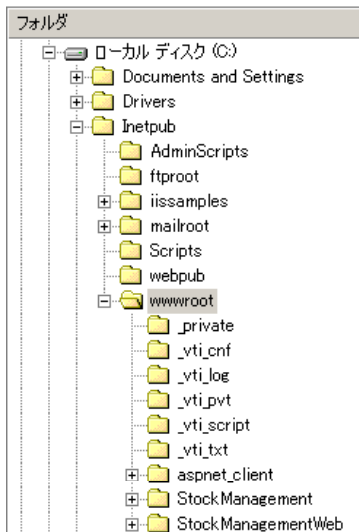
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options
に接続されました。
SQL> @install.sql
```

画面 3 scott ユーザーでテーブル・パッケージ作成スクリプトを実行する画面

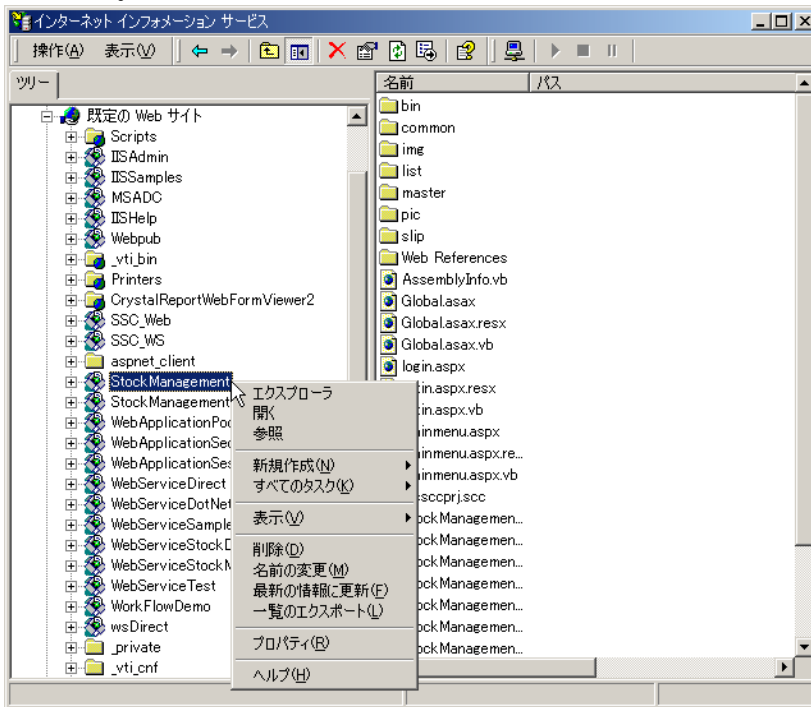
メモ： 上記のサンプルでは SCOTT ユーザーでログインしていますが、任意のユーザー名でログインしてください。また、1の作業で作成したディレクトリオブジェクトを削除したい場合は、SQL*Plus に system ユーザーでログインしなおし、drop directory imkdir; と入力してください。

Internet Information Services(IIS)の設定 (アプリケーション・サーバー上での設定)

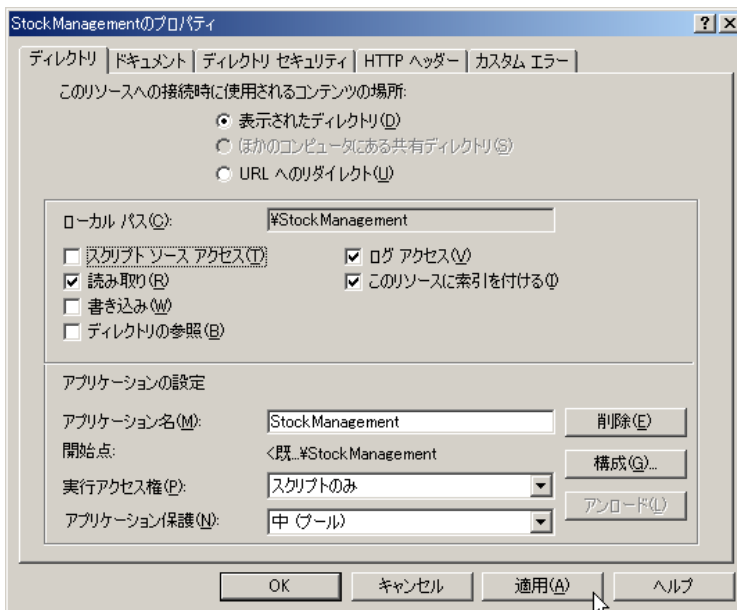
ダウンロードファイル (sample.zip) を解凍後に作成される StotckManagement というディレクトリと、WebServiceStockManagement というディレクトリを Web サーバー上の Internet Information Services(IIS)の root ディレクトリ下にディレクトリごとコピーしてください。コピー後は以下のようなディレクトリ構成になります。



コピーが終わりましたら、「コントロール・パネル」の管理ツールより「インターネット・サービス・マネージャ」を起動してください。StockManagement フォルダの設定を変更します。Internet Information Services(IIS)の管理画面を開き、以下のように StockManagement フォルダを右クリックしてください。



メニューのプロパティを選択しますと、以下のプロパティ設定の画面が表示されます。



上記の画面で「アプリケーションの設定項目」にある「作成」ボタンをクリックするとアプリケーションの設定が行われます。作成されたら「OK」をクリックしてプロパティ・ウィンドウを終了してください。同様の作業を `WebServiceStockManagement` フォルダにも行ってください。

Visual Studio .NET でソリューションを開く (開発クライアント上での設定)

Internet Information Services(IIS)の設定で `StockManagement` フォルダを公開設定したと思いますが、そのフォルダ内にある `StockManagement.sln` ファイル (Visual Studio .NET のソリューションファイル) をダブルクリックでしますと、Visual Studio .NET が起動されます。ソリューション・エクスプローラーには以下の2つのプロジェクトが開かれた状態になっております。



2つのプロジェクトはそれぞれ以下のような構成になっております。

- `StockManagement`
ASP.NET Web アプリケーション側： WebForm アプリケーション。
- `WebServiceStockManagement`
Web サービス側： XML Web サービスアプリケーション。

メモ：データアクセス部分は `WebServiceStockManagement` プロジェクトの `ServiceStockManagement.asmx` に記述されています。

上記の設定で必要なファイルがインストールされ、実行可能になります。これから実際に ODP.NET を使用したアプリケーション開発の方法をサンプル・アプリケーションのコードを参照しながら説明します。

ASP.NET からの LOB の扱い

今回のサンプルアプリケーションでは、商品画像を Oracle のバイナリデータを格納するための型、「BLOB 型」として保存しています。実際に、ASP.NET から ODP.NET を使用して BLOB フィールドを扱うための方法を説明します。

LOB フィールドへの値の格納

画像ファイルを ASP.NET の WEB フォームから選択させ、Oracle の BLOB 型フィールドへの格納は以下のようになります。

1. ASPX ファイルの `<form>` タグの修正。

ENCTYPE 属性では、送信されるデータの形式を指定します。ブラウザは ENCTYPE 属性を使用して、サーバーに送信される情報をエンコードします。このコードの `action` 属性によって、ページが要求を処理するように指定されます。

<ソース `product.aspx` - 「`<form>`タグ」指定部>

```
<form id="Form1" method="post" enctype="multipart/form-data" runat="server">
```

2. サーバーにアップロードするファイルを指定するための **Input** コントロールを追加します。

<ソース product.aspx>

```
<INPUT id="FileImage" type="file" size="52" name="FileImage" runat="server">
```

3. サーバーにアップロードされたファイルを変数に格納するための宣言と設定を行います。

<ソース product.aspx.vb – 変数宣言部と、Page_Load イベント部>

```

master*product.aspx.vb
product
Public Class product
    Inherits System.Web.UI.Page

    Private myFile As HttpPostedFile

    Web フォーム デザイナで生成されたコード
    Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        ページを初期化するユーザー コードをここに挿入します。
        If Request.Files.Count > 0 Then
            myFile = Request.Files(0)
        End If

        If Not IsPostBack Then
            setCmb()

            If Request.QueryString("productcode") <> "" Then
                TextBoxProductCode.Text = Request.QueryString("productcode")
                GetProductData()
            End If
        End If
    End Sub

    ' Webサービスを呼び出して 商品情報をフォームに表示します。
  
```

変数宣言部では、Post された画像ファイルを格納するためのクラス、HttpPostedFile クラスを宣言しています。

```
Private myFile As HttpPostedFile
```

Page_Load イベントで、Post された値を変数に格納しています。

```

If Request.Files.Count > 0 Then
    myFile = Request.Files(0)
End If
  
```

4. Post された画像ファイルを **Byte** 変数に格納し、WEB サービスに引数として渡します。

<ソース product.aspx.vb – UpdateProductData メソッド>

```
// 画像ファイルの格納
Dim FileLen As Integer = myFile.ContentLength
Dim imgByte(FileLen) As Byte

' Initialize the stream.
Dim MyStream As System.IO.Stream = myFile.InputStream

' Read the file into the byte array.
MyStream.Read(imgByte, 0, FileLen)

// XML WEB サービス側で商品情報を更新
Dim wSvc As New ws_sman.ServiceStockManagement
Try
    wSvc.UpdateProductData(TextBoxProductCode.Text, _
        TextBoxProductName.Text, _
        CInt(TextBoxPrice.Text), _
        CInt(TextBoxStockNum.Text), _
        TextBoxUnit.Text, _
        TextBoxRemarks.Text, _
        DropDownListSupplierCode.SelectedValue, _
        CInt(TextBoxPurchasePrice.Text), _
        CInt(TextBoxOrderPoint.Text), _
        imgByte)
~ 以下省略 ~
```

5. WEB サービス側で Byte 配列から BLOB フィールドへの格納

WEB サービス側でイメージファイルを Byte 配列として受信し、BLOB フィールドへ更新を行います。

<ソース ServiceStockManager.asmx.vb - UpdateProductData メソッド>

```
<WebMethod(TransactionOption:=TransactionOption.RequiresNew)> _
Public Sub UpdateProductData(ByVal sProductCode As String, _
    ~ 省略 ~
    ByVal byteImage() As Byte)

    Dim prm(10) As OracleParameter

    DbOpen()
    sSql = "begin " & _
        ~ 省略 ~
        "pkg_product.ProductRec.productimage := :productimage; " & _
        "pkg_product.UpdateProductData; " & _
        "end; "
    cmd.CommandText = sSql
    ~ 省略 ~
    prm(9) = cmd.Parameters.Add("productimage", byteImage)
    prm(9).OracleDbType = OracleDbType.Blob
    cmd.ExecuteNonQuery()

    cnn.Close()
End Sub
```

LOB フィールドから値を取得

BLOB フィールドから画像データを取得し、ASP.NET の WEB フォームで表示する方法について説明します。

1. WEB サービス側で商品情報を DataSet オブジェクトに格納

<ソース ServiceStockManager.asmx.vb - GetProductData メソッド>

```
sSql = "select prod.productcode," & _  
    ~ 省略 ~  
    "prod.productimage " & _  
    "from " & _  
    "product prod, supplier suppli " & _  
    "where " & _  
    "prod.suppliercode = suppli.suppliercode and " & _  
    "prod.productcode = :ProductCode"  
cmd.CommandText = sSql  
Dim pProductCode As OracleParameter = _  
    cmd.Parameters.Add("ProductCode", sProductCode)  
Dim da As New OracleDataAdapter(cmd)  
If dsData.Tables.Contains("ProductData") Then dsData.Tables("ProductData").Clear()  
da.Fill(dsData, "ProductData")
```

2. ASP.NET の WEB フォーム上で DataSet オブジェクトの画像データを表示

<ソース ServiceStockManager.asmx.vb - UpdateProductData メソッド>

```
Dim wSvc As New ws_sman.ServiceStockManagement
Dim dsProductData As New DataSet

Try
    wSvc.GetProductData(Request.QueryString("productcode"), dsProductData)

    '//パラメータの設定
    Dim byteImg() As Byte = _
        dsProductData.Tables("ProductData").Rows(0).Item("productimage")
    Response.ContentType = "image/jpeg"
    Response.BinaryWrite(byteImg)
    Response.End()
Catch ex As Exception
    'MessageBox.Show(ex.Message)
Finally
    wSvc.Dispose()
End Try
```

ASP.NET アプリケーションのセキュリティについて

今回のサンプルアプリケーションでは、ユーザー認証に ASP.NET の Forms 認証を行っております。(ASP.NET の Forms 認証についての詳細については、Microsoft 社のサイトを参照してください。) 認証によるセキュリティの確保以外にも、どのユーザーがどのような操作を行ったかを監査したい場合もあります。その場合、ASP.NET もしくは、XML WEB サービスを利用した 3 層アプリケーションでは、普通であれば、中間層から共通のユーザーでデータベースへ接続しますが、その場合、個々のユーザー毎でのデータベースのアクセス制御や監査は行えないと思います。3 層アプリケーションで監査を実行するにはプロキシ認証を行う方法と、アプリケーション・コンテキストの利用する方法があります。

プロキシ認証

プロキシ認証を利用すると、接続のためのユーザーは共有するが、データベースのアクセス権等は、個々のユーザー毎で実施することが可能になります。

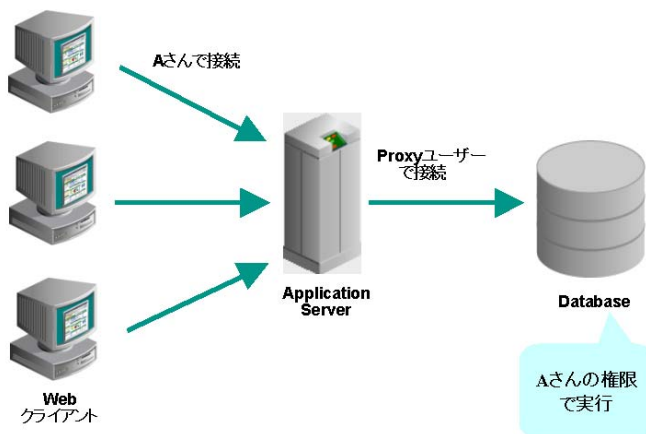


図 1 PROXY 認証での接続例

プロキシ認証の設定は、OracleConnection クラスの Connection プロパティで設定します。

```

Dim cnn As New OracleConnection

cnn.ConnectionString = _
    "User Id=scott;Password=tiger;Data Source=orcl;" + _
    "Proxy User Id=staffuser;Proxy Password=staffpassword"
cnn.Open()
  
```

プロキシ認証を利用した接続

プロキシ認証を行うことによりデータベースの権限をアプリケーションで享受可能になります。その結果プロキシ認証で指定したユーザーでなければデータへのアクセス権限がないので、より安全な接続環境を実現できます。また、DB ユーザー毎に監査を行うことが可能になります。Oracle のセキュリティに関する情報は、OTN(Oracle Technology Network)の「Documentation Library」にて詳細な手順を説明していますのでそちらを参照してください。今回のサンプルでは staff ユーザーでアクセスする際に、Oracle のエラーがそのまま WEB フォーム上に表示されてしまいましたが、実際のアプリケーションではエラー情報を取得して適切に対処する必要があります。その場合は、OracleException クラスを利用します。OracleException を利用したエラーハンドリングは後述しますが、詳細は 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。

クライアント識別子の利用

クライアント識別子を使用することにより、以下のことが可能になります。

- 同一DBユーザーでセッション毎に異なるアクセス権を設定可能
- SYSTEM ユーザーや表の持主のアクセスも制御可

アプリケーション・コンテキストを使用すると、セッション毎に固有の属性値を格納できます。

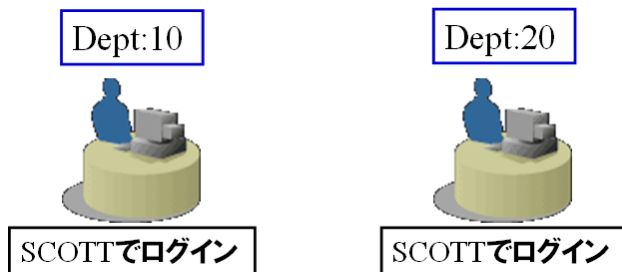


図2 セッション毎に固有の属性値を格納

今回のサンプルアプリケーションでは、得意先マスタの検索時にどのユーザーが検索を行ったかを監査できるように、クライアント識別子を設定しています。クライアント識別子の設定は、`exec DBMS_SESSION.SET_IDENTIFIER('ユーザー名');`

上記の SQL を実行すると、監査用テーブル `dba_audit_trail` の `CLIENT_ID` 列に値が設定されます。

<ソース ServiceStockManagement.aspx- GetCustomerListAudit メソッド>

```
cmd.CommandType = CommandType.StoredProcedure
cmd.CommandText = "DBMS_SESSION.SET_IDENTIFIER"
cmd.Parameters.Add(New OracleParameter("pUserId", sUserId))
cmd.ExecuteNonQuery()
cmd.Parameters.Clear()
```

セッション識別子にユーザーの属性を設定すると、監査ログの CLIENT_ID にクライアント識別子が設定されます。

```
SQL> select timestamp, client_id, sql_text from dba_audit_trail;

TIMESTAMP CLIENT_ID SQL_TEXT
-----
04-12-21 T0000001 select * from customer
04-12-21 T0000002 select * from customer where ~
04-12-21 T0000109 select customername from customer
```

クライアント識別子

上記のように監査以外にも、Oracle 10g Enterprise Edition ではファイングレイン・アクセスコントロールにより、アプリケーションを修正せずに、Oracle Database 上の設定で、データアクセスの制御を行えます。

ファイングレイン・アクセスコントロール

- 自動的にWHERE句を付与
 - Oracle Database 10gより列名の制限も可

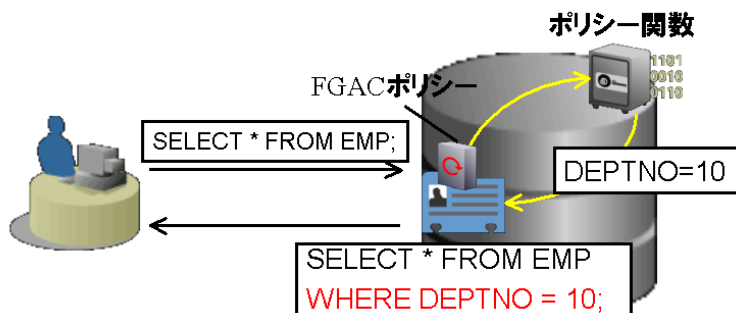


図3 ファイングレイン・アクセスコントロールによるデータアクセス制御

メモ：ファイングレインアクセスコントロールの具体的な実装例などはマニュアルをご覧ください。

「Oracle 8i アプリケーション開発者ガイド 基礎編」 P11-53(8.1.7 の場合)

「Oracle 9i アプリケーション開発者ガイド 基礎編」 P12-15(9.2.0 の場合)

「Oracle Database セキュリティガイド 10g リリース 1」 P14-7

マニュアルは OTN よりダウンロード可能です

<http://otn.oracle.co.jp/document/index.html>

ODP.NET でのエラーハンドリング

OracleException クラスは、Oracle Data Provider for .NET でエラーが発生した場合にスローされる例外を表します。各 OracleException オブジェクトには、エラーまたは警告を説明する Error プロパティの OracleError オブジェクトが 1 つ以上含まれています。OracleException のエラーハンドリングを実装したコードは以下のようになります。

<ソース ServiceStockManagement.asmx- UpdateOrderData メソッド>

```
Try
    cmd.ExecuteNonQuery()
    retVal = prm(6).Value
Catch ex As OracleException
    WriteEventLog(ex)
    retVal = 0
    ContextUtil.SetAbort()
End Try
```

上記のコードでは WriteEventLog というメソッドをコールし、エラーの内容をイベントログに出力しています。WriteEventLog のコードは以下のようになります。

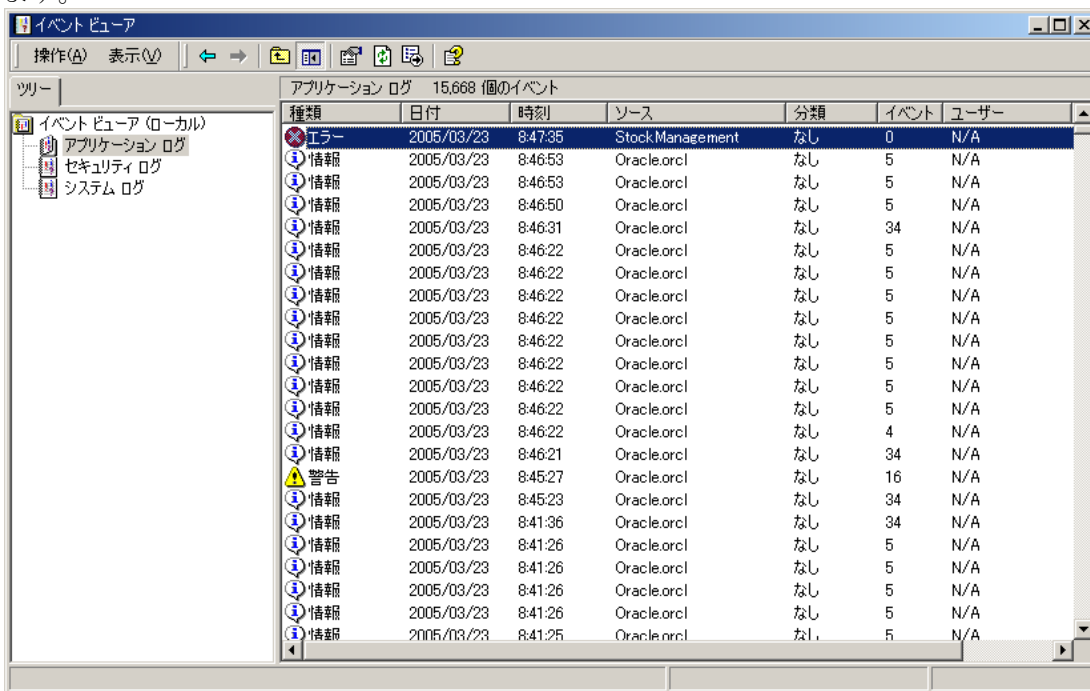
<ソース ServiceStockManagement.asmx- WriteEventLog メソッド>

```
Private Sub WriteEventLog(ByVal ex As OracleException)
    Dim appLog As EventLog = New EventLog("Application")

    appLog.Source = "StockManagement"
    appLog.WriteEntry(ex.Message, EventLogEntryType.Error)

    appLog.Close()
    appLog.Dispose()
End Sub
```

以上の WriteEventLog メソッドをでは、Windows のイベントビューアー上にエラーの内容を追記しております。「appLog.Source = "StockManagement"」の部分で、イベントビューアーに表示されるソースになります。



画面 4 イベントビューアー上でのエラーの確認

ASPNET ユーザーはイベントログにイベントソースを作成するための適切なユーザー権限を持っていないので、以下の手順でレジストリエディタを使用し、Application イベントログの下にイベントソースを作成します。

1. [スタート] ボタンをクリックし、[ファイル名を指定して実行] をクリックします。
2. [名前] ボックスに `regedit` と入力します。
3. 次のレジストリ サブキーに移動します。

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog\Application

4. Application サブキーを右クリックし、[新規] をポイントして、[キー] をクリックします。
5. キー名として `StockManagement` と入力します。（この値は EventLog オブジェクトの Source プロパティで設定する、イベントビューアーに表示されるソースの値になります。）
6. レジストリ エディタを閉じます。

OracleException で提供される、メンバー/メソッド/プロパティの一覧は、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。

ODP.NET と Web サーバーコントロールの連携

ASP.NET では多数の Web サーバーコントロールが用意されています。Visual Studio.NET を使用して、ASP.NET Web アプリケーションを開発する際には、Web フォーム上にツールボックスからコントロールを配置します。Web サーバー上で実行されるコードは、配置された各コントロールのプロパティにアクセスして処理を行うことができます。実際に ODP.NET と Web サーバーコントロールの連携方法について説明します。

型付データセットの作成

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。データバインドを利用して、Web サーバーコントロールと連携することにより以下のメリットがあります。

- コードの可読性の向上
- コードの信頼性の向上

例えば、通常のデータセットを利用した場合と型付データセットを利用した場合のコードは以下のようになります。

```
Dim strCustomerId = dsCustomer.Tables("MSTCUSTOMER").Rows(5).Item("CUSTOMERID")
If (dsCustomer.Tables("MSTCUSTOMER").Rows(5).Item("CUSTOMERID") Is DBNull.Value) Then
    'Null 時の処理を記述
End If
```

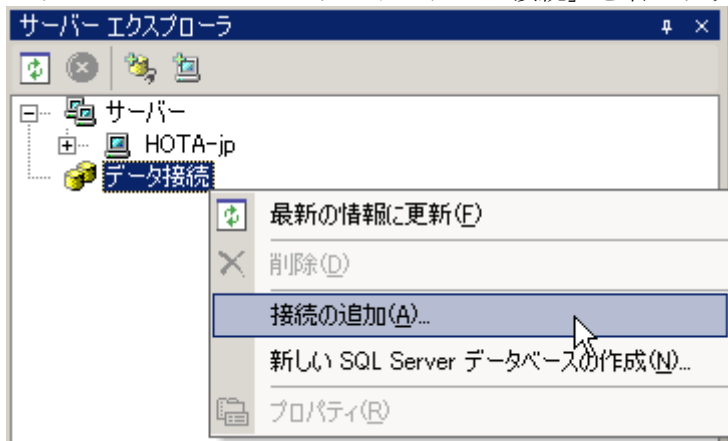
型なしデータセットを使用した場合

```
Dim strCustomerId = dsCustomer.MSTCUSTOMER(5).CUSTOMERID
If (dsCustomer.MSTCUSTOMER(5).IsADDRESSNull) Then
    'Null 時の処理を記述
End If
```

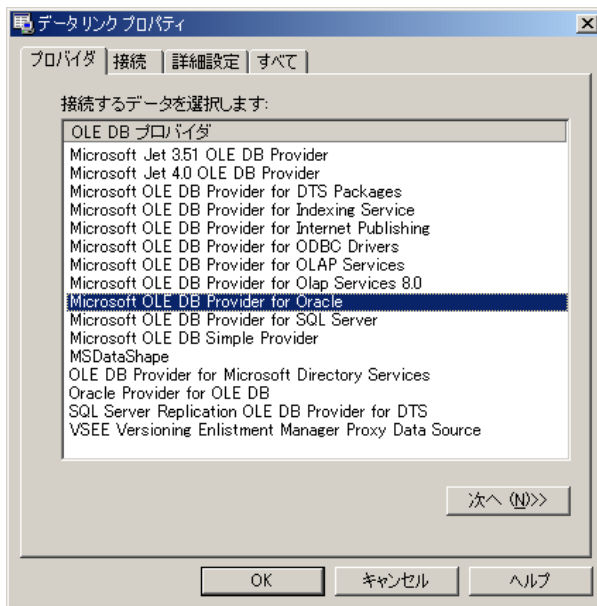
型付データセットを使用した場合

上記サンプルコードのようにソースの可読性がよくなるだけでなく、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したりすることも可能です。では、実際に **Oracle** で管理されているテーブル情報をもとに型付データセットを作成してみましょう。

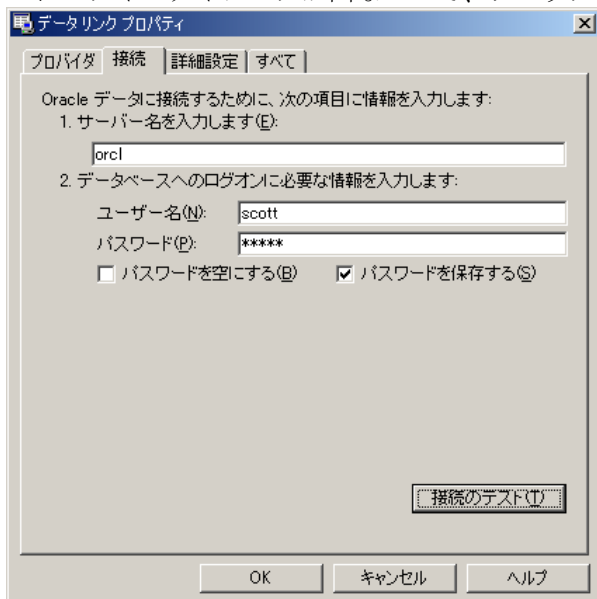
1. サーバーエクスプローラより「データ接続」を右クリックし、「接続の追加」をクリック



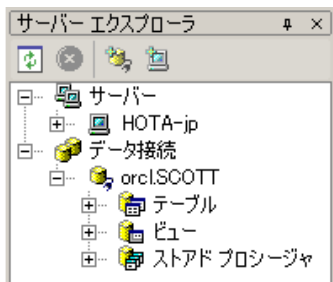
2. データリンクプロパティが表示されるので、「Microsoft OLE DB Provider for Oracle」を選択し、「次へ」ボタンをクリック



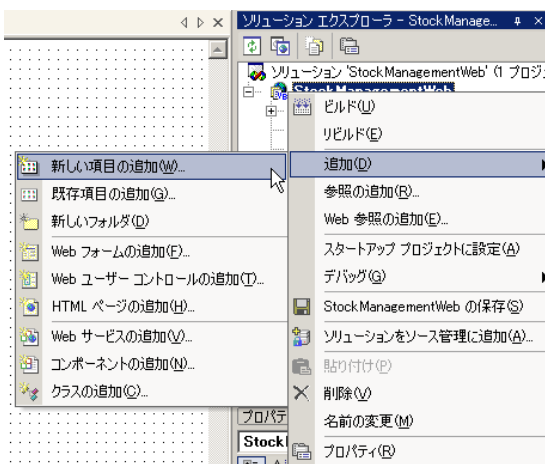
3. サーバー名にネットサービス名を入力。ユーザー名とパスワードは Oracle ユーザーとパスワードを入力。また、「パスワードを保存する」チェックボックスにチェックを入れておくと、接続の度にログインダイアログが出ないので、チェックをしていただくことをお勧めします。



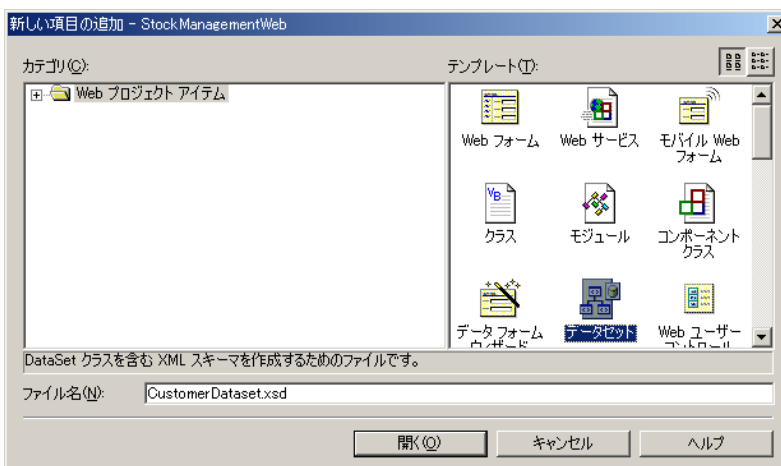
4. 3の「接続のテスト」を実行し、接続の確認後に「OK」ボタンをクリックしてデータリンクプロパティを閉じてください。サーバーエクスプローラのデータ接続に設定後の接続が追加されているのが確認できます。



- ソリューションエクスプローラよりプロジェクトを右クリックし、「追加」→「新しい項目の追加」を選択してください。

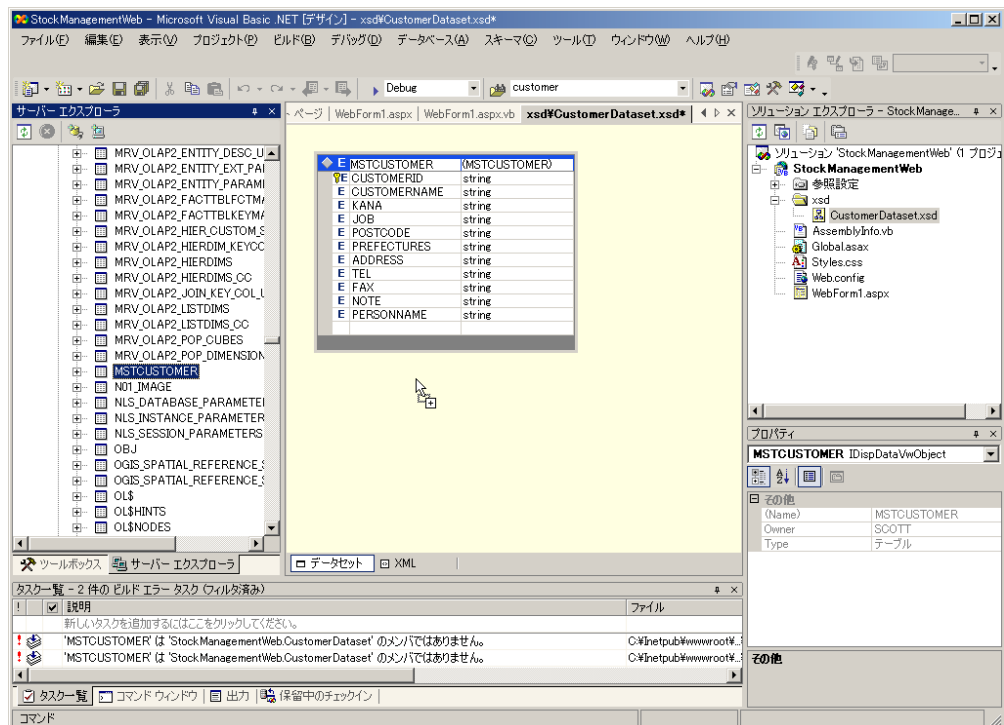


- 新しい項目の追加ウィンドウでデータセットを選択し、任意のファイル名を入力して「開く」ボタンをクリックしてください。

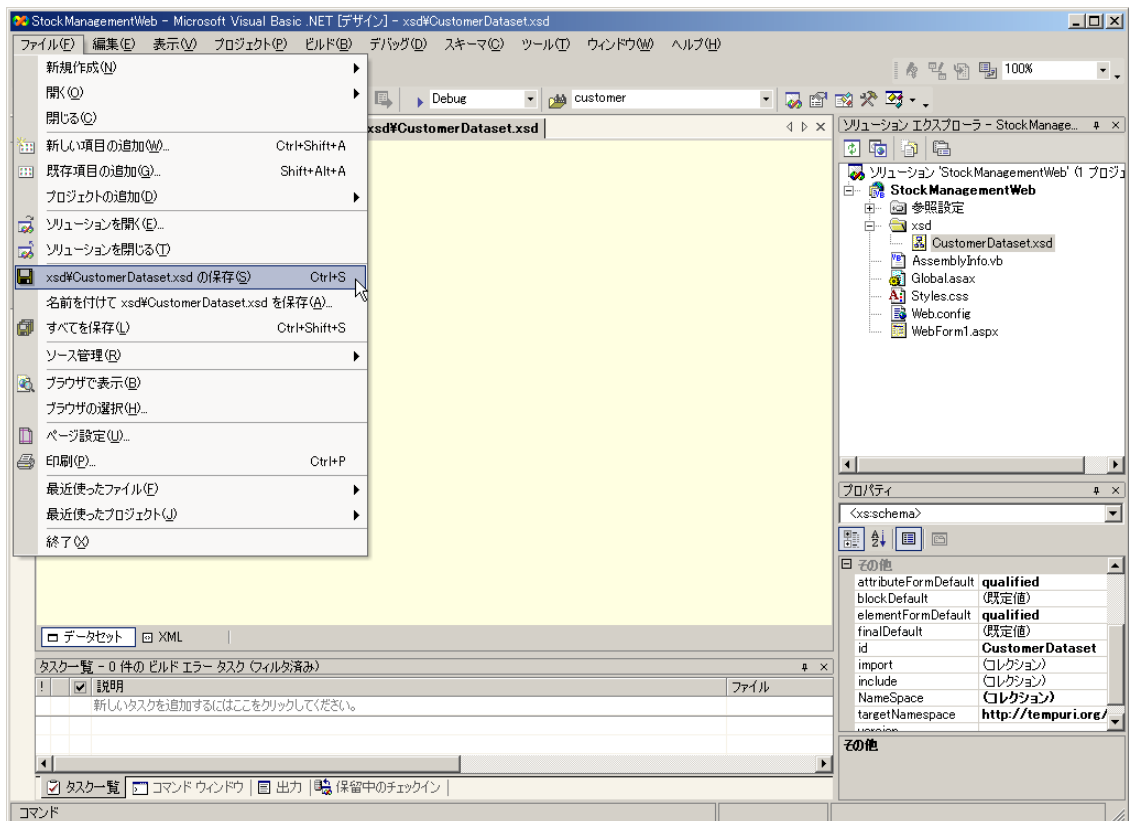


- データセットのデザイン画面が開きますので、サーバーエクスプローラより型付データセットを作成したい対象のテーブルを選択して、ドラッグアンドドロップでデザイン画面に貼り

付けてください。



- 「ファイル」メニューより作成されたデータセット(xsd ファイル)を保存すると型付データセットが作成されます。



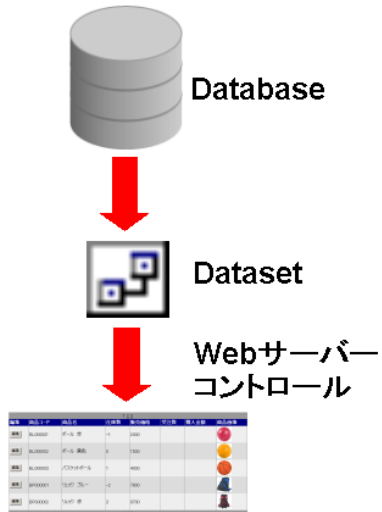
9. 作成された型付データセットはクラスとして作成されます。VB.NET から型付データセットを利用するには、以下のように型付データセットのインスタンスを生成します。

```
Dim dsCustomer As New CustomerDataset
```

型付データセットのインスタンスを生成

Web サーバーコントロールの連携

Oracle データベースからデータを取得し、Web サーバーコントロールと連携する方法として、Dataset オブジェクトを使用します。Oracle データベースの値を一度 Dataset オブジェクトに格納して、データバインディング機能を使用して、Web サーバーコントロールに結果を表示します。



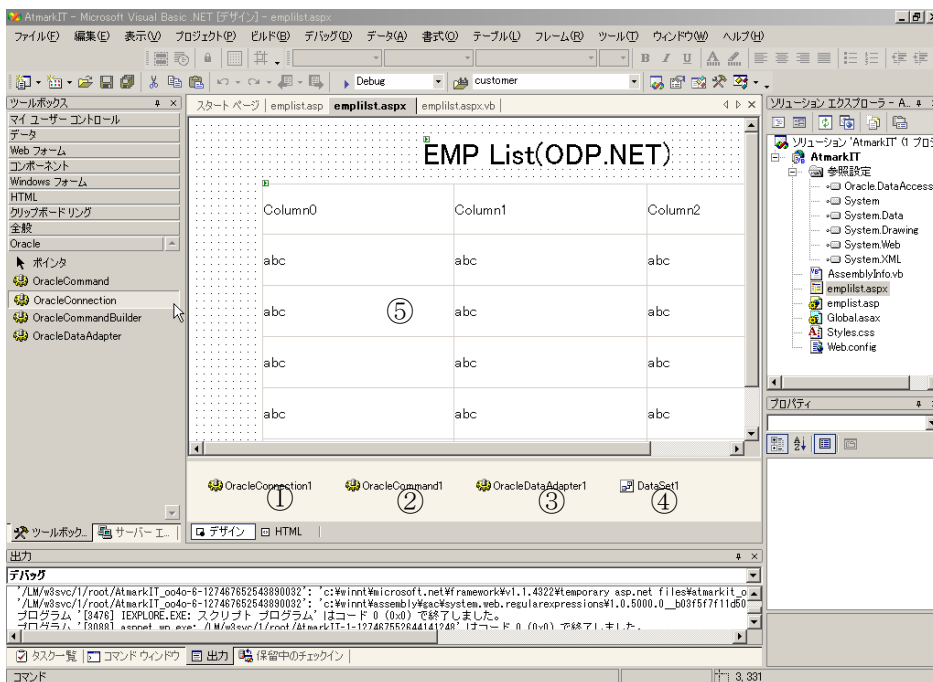
先ほど作成した型付データセットを利用して、Dataset と Web サーバーコントロールの連携方法を説明します。

1. WEB フォームの設定

ASP.NET の WEB フォーム上に以下のコントロールを貼り付けます。

- OracleConnection
- OracleCommand
- OracleDataAdapter
- DataSet
- DataGrid

ODP.NETのコントロールをツールボックスに追加するには、OTN(Oracle Technology Network)の「[意外と簡単!?.NETでOracle](#)」にて詳細な手順を説明していますのでそちらを参照してください



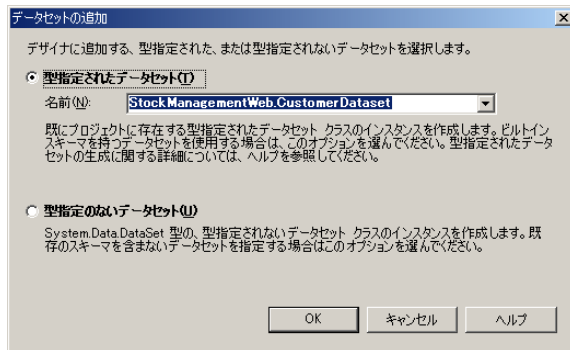
2. コントロールのプロパティの値を設定

Web フォーム上に貼り付けたコントロールに対してプロパティの値を以下のように設定します。

コントロール	コントロール名	プロパティ	設定値
①OracleConnection	OracleConnection1	ConnectionString	User Id=scott;Password=tiger;Data Source=orcl (orcl の部分はサービス名を指定してください。)
②OracleCommand	OracleCommand1	Connection	OracleConnection1 を指定
③OracleDataAdapter	OracleDataAdapter1	SelectCommand	OracleCommand1 を指定
④CustomerDataset	CustomerDataset1	変更なし	変更なし
⑤DataGrid	DataGrid1	変更なし	変更なし

表 2 ODP.NET で提供されているオブジェクト一覧

上記のコントロールを Web フォーム上に貼り付ける際に、DataSet コントロールを Web フォームに貼り付けると、以下の「データセットの追加」ウィンドウが表示されます。この時に、「型指定されたデータセット」を選択し、既に作成された型付データセットをします。



3. Page_Load イベントにコードを記述

Page_Load イベントに以下のコードを記述します。

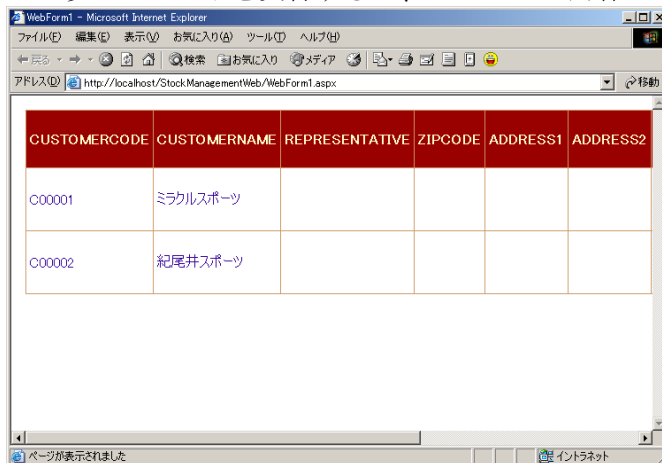
```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load
```

```
    OracleCommand1.CommandText = "select * from mstcustomer order by customerid"
    OracleDataAdapter1.Fill(CustomerDataSet1, "CustomerList1")
    DataGrid1.DataSource = CustomerDataSet1
    DataGrid1.DataMember = " CustomerList1"
    DataBind()
```

```
End Sub
```

データバインドを使用して DataGrid へ emp 表を表示するサンプルコード(VB.NET)

4. 以上のコードを実行すると、テーブルの内容が DataGrid コントロールに表示されます。

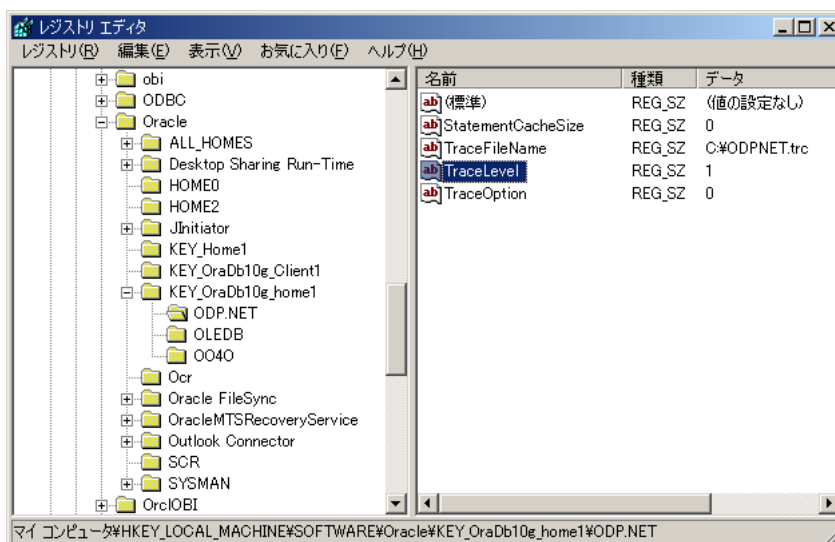


ODP.NET のデバッグ・トレース機能の利用

ODP.NET では、すべての ODP.NET アクティビティをトレース・ファイルにログングできるデバッグ・トレースがサポートされています。ASP.NET のような Web アプリケーションでは複数のユーザーが接続プールを利用して同一接続から SQL を発行するような場面が多いと思います。そのような場合、接続プールの状況をトレース・ファイルで確認が取れます。様々なレベルのトレースが可能で、トレースレベルの設定はレジストリで行います。

デバッグ・トレースのためのレジストリ設定

レジストリ設定は、次のディレクトリの下に構成してください。
HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOMEID\ODP.NET\HOME HOMEID は該当する Oracle ホーム・ディレクトリです。



トレースレベルの設定

上記のレジストリ設定の TraceLevel を設定することによりさまざまなレベルのトレース情報を取得することが可能です。

トレースレベル	説明
0	なし
1	入力、出力および SQL 実行情報
2	接続プーリング統計
4	分散トランザクション（登録および登録解除）

複数のオブジェクトについてのトレースを取得するには、有効な値を追加するだけです。たとえば、TraceLevel が 3 に設定されている場合、トレース情報は入力、出力、SQL および接続プーリング情報についてロギングされます。

トレースオプションの設定

TraceOption の有効な値は、次のとおりです。

オプション	説明
0	1 つのトレース・ファイル
1	複数のトレース・ファイル

TraceOption は、トレース情報を 1 つのファイルに記録するか、スレッド ID ごとの複数のファイルに記録するかを指定します。単一トレース・ファイルを指定した場合は、TraceFileName に指定されたファイル名が使用されます。複数のトレース・ファイル・オプションを要求した場合、スレッドごとにトレース・ファイルを作成するために、指定されているファイル名にスレッド ID が追加されます。

文キャッシングについて

ASP.NET などの WEB アプリケーションでは、通常は、アプリケーション・サーバー等の中間層からデータベースへのアクセスを共有します。その場合、文キャッシングを利用することにより、同一のコネクションで発行される同じ SQL もしくは PL/SQL 文を後で実行した際、カーソルから解析された情報を再利用して実行することでパフォーマンスが向上します。文キャッシングを有効にするためには接続文字列属性に「Statement Cache Size」を指定することにより可能になります。「Statement Cache Size」で指定する値は、実際にキャッシュする文の最大数を指定します。以下の接続文字列属性では最大 200 個の文をキャッシュする設定を行っております。

<ソース ServiceStockManagement.aspx- DbConnect メソッド>

```
cnn.ConnectionString = _
"user id=scott;password=tiger;data source=orcl;Statement Cache Size=200"
```

デ
フォ

ルトでは、この属性は 0 に設定され、文キャッシングは無効です。指定された最大キャッシュ・サイズまで文がキャッシュされると、使用頻度のもっとも低い文が解放され、新しく作成された文をキャッシュするためのスペースが確保されます。キャッシュされた文を全て消去するためには、OracleConnection の PurgeStatementCache メソッド をコールします。また、文キャッシングを有効にするためには、SQL 文または PL/SQL 文はリテラル値ではなくパラメータを使用する必要があります。例えばテキストボックスの値をパラメータとして設定しているコードは以下のようになります。

```
Dim cmd As New OracleCommand  
  
cmd.CommandText = "Select * from emp where ename like :ename"  
  
Dim paraEname As OracleParameter = cmd.Parameters.Add("ename",  
TextboxEname.Text + "%")
```

パラメータを使用することにより文キャッシングが有効になるコード (VB.NET)

以下のコードようにリテラル値で SQL または PL/SQL 文を指定すると文キャッシングが有効になりません。

```
Dim cmd As New OracleCommand  
  
cmd.CommandText = "Select * from emp where ename like " +  
TextboxEname.Text + "%"
```

文キャッシングが有効にならないコード。(VB.NET)

最新のODP.NETのダウンロードは、[OTNのホームページ](#)から、「ソフトウェアダウンロード」のリンクを選択し、「テクノロジー/ユーティリティ/ドライバ」の欄から、「Oracle Data Provider for .NET(ODP.NET)」のリンクをクリックしてください。また、文キャッシングについては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。

グローバリゼーションのサポート

最近の Web アプリケーションは、国内の使用のみならず複数の国をまたがったユーザーの使用も多くなってきています。ODP.NET のグローバリゼーション・サポートによって、アプリケーションで、文化によって異なるデータを適切に操作できます。この機能を使用すると、Oracle グローバリゼーション設定に従って適切な文字列フォーマット、日付、時刻、通貨、数値、ソート順序およびカレンダー規則が適用されます。

グローバリゼーションの種類

ODP.NET では以下の 3 種類のグローバリゼーションをサポートします。

- クライアント・グローバリゼーション
- セッション・グローバリゼーション
- スレッドベースのグローバリゼーション

- クライアント・グローバリゼーション

クライアント・グローバリゼーション設定は、ローカル・コンピュータの Windows レジストリ内の Oracle グローバリゼーション設定 (NLS_ANG) から導出されます。クライアント・グローバリゼーション・パラメータ設定は読み取り専用であり、アプリケーションの存続期間中は変わりません。クライアント・グローバリゼーション設定は、OracleGlobalization.GetClientInfo メソッドをコールすることにより取得できます。次のサンプルコードでは、クライアント・グローバリゼーション設定を取得しています。

```
Dim ClientGlob OracleGlobalization = OracleGlobalization.GetClientInfo()
```

- セッション・グローバリゼーション

セッション・グローバリゼーション・パラメータは、最初はクライアント・グローバリゼーション設定と同じです。クライアント設定とは異なり、セッション・グローバリゼーション設定は更新可能です。ただし、セッション・グローバリゼーション設定はデータベース・サーバーへの接続を確立した後でしか取得できません。セッション・グローバリゼーション設定は、OracleConnection の GetSessionInfo() をコールすることにより取得できます。このメソッドをコールすると、セッションのグローバリゼーション設定を表すプロパティを持つ OracleGlobalization オブジェクトのインスタンスが返されます。OracleConnection オブジェクトにより、接続が確立されるたびに、クライアント・コンピュータの Oracle グローバリゼーション (または National Language Support (NLS)) レジストリ設定で指定されている値で初期化されたグローバリゼーション・パラメータを持つセッションが暗黙的にオープンされます。セッションの設定は更新可能であり、その存続期間中も変更できます。次のサンプルコードでは、セッションの日付書

式設定を変更します。

```
Dim cnn As New OracleConnection("user id=scott;password=tiger;data source=orcl")
Dim cmd As New OracleCommand
cmd.Connection = cnn
cnn.Open()
Dim info As OracleGlobalization = cnn.GetSessionInfo
info.DateFormat = "yyyy/mm/dd"
cnn.SetSessionInfo(info)
```

- スレッドベースのグローバリゼーション

スレッドベースのグローバリゼーション・パラメータ設定は、各スレッドに固有です。これらの設定は、最初はクライアント・グローバリゼーション・パラメータと同じですが、アプリケーションで指定されたとおりに変更できます。ODP.NET 型と文字列との間で変換が行われるとき、適用可能であればスレッドベースのグローバリゼーション・パラメータが使用されます。スレッドベースのグローバリゼーション・パラメータ設定を取得するには、OracleGlobalization オブジェクトの `GetThreadInfo` static メソッドをコールします。スレッドのグローバリゼーション設定を行うには、OracleGlobalization オブジェクトの `SetThreadInfo` static メソッドをコールできます。文化によって異なるデータを操作する場合、ODP.NET のクラスおよび構造体は OracleGlobalization 設定のみに依存します。.NET でのスレッドベースのグローバリゼーション情報は使用されません。また、アプリケーションで .NET 型しか使用されない場合、OracleGlobalization 設定は影響しません。しかし、ODP.NET 型と .NET 型の間で変換が行われるときは、適用可能であれば OracleGlobalization 設定が使用されます。

`System.Threading.Thread.CurrentThread.CurrentCulture`が変更されても、スレッドまたはセッションのOracleGlobalizationの設定に影響はありません。またその逆も同様です。

次のコードは、ODP.NET 型によってスレッドのグローバリゼーション設定がどのように使用されるかを示しています。

```
Dim ThreadGlob As OracleGlobalization = OracleGlobalization.GetThreadInfo()
ThreadGlob.DateFormat = "YYYY-MM-DD"
OracleGlobalization.SetThreadInfo(ThreadGlob)
Dim OraDate As OracleDate = New OracleDate("2002-01-01")
```

OracleGlobalization オブジェクトでは、そのプロパティに対して行われた変更が検証されます。プロパティの設定に無効な値が使用されている場合、例外がスローされます。Territory および Language プロパティが変更されると、OracleGlobalization オブジェクトの他のプロパティが暗黙的に変更されますので注意してください。

グローバリゼーションの影響を受ける操作

この項では、グローバリゼーション設定に依存する、またはこの設定の影響を受ける ODP.NET 型および操作をリストします。

- クライアント・コンピュータのグローバリゼーション設定に依存する操作

OracleString 構造体は、クライアント・コンピュータの OracleGlobalization 設定によって決まります。GetNonUnicode メソッドで Unicode 文字列を byte[] に変換する場合、および byte[] を受け入れる OracleString コンストラクタで ANSI 文字の byte[] を Unicode に変換する場合、ローカル・コンピュータのクライアント・キャラクタ・セットが使用されます。
- スレッド・グローバリゼーション設定に依存する操作

OracleString 構造体は、クライアント・コンピュータの OracleGlobalization 設定によって決まります。GetNonUnicode メソッドで Unicode 文字列を byte[] に変換する場合、および byte[] を受け入れる OracleString コンストラクタで ANSI 文字の byte[] を Unicode に変換する場合、ローカル・コンピュータのクライアント・キャラクタ・セットが使用されます。
- セッション・グローバリゼーション・パラメータの影響を受ける操作

セッション・グローバリゼーション設定は、文字列としてサーバーから取得される、またはサーバーに送信されるすべてのデータに影響を与えます。たとえば、TO_CHAR()関数が適用された DATE 列が選択されている場合、DATE 列のデータは、セッション・グローバリゼーション設定の DateFormat で指定された日付書式の文字列になります。逆の方向にデータを送信するには、DATE 列に挿入する文字列データを、セッション・グローバリゼーション設定の DateFormat プロパティで指定された書式にする必要があります。また、セッション・グローバリゼーション設定は、安全な型マッピングを使用して文字列として DataSet 内に取得されるデー

タにも影響を与えます。型の書式が区別される場合、文字列は常に、セッション・グローバル化セッション設定で指定されている書式になります。

たとえば、VARCHAR2 および CHAR データは、適用可能な書式がないため、セッション設定の影響を受けません。しかし、DateFormat および NumericCharacters プロパティは、DATE 型と NUMBER 型が安全な型マッピングを使用してデータベース・サーバーから文字列として取得された場合、それぞれの文字列表現に影響を与える可能性があります。



日本オラクル株式会社

Copyright © 2005 Oracle. All rights reserved.

このドキュメントは単に情報として提供され、内容は予告なしに変更される場合があります。このドキュメントに誤りが無いことの保証や、商品性又は特定目的への適合性の黙示的な保証や条件を含め明示的又は黙示的な保証や条件は一切無いものとします。オラクル社は、このドキュメントについていかなる責任も負いません。また、このドキュメントによって直接又は間接にいかなる契約上の義務も負うものではありません。このドキュメントを形式、手段（電子的又は機械的）、目的に関係なく、オラクル社の書面による事前の承諾なく、複製又は転載することはできません。

Oracle は Oracle Corporation およびその関連企業の登録商標です。その他の名称は、各社の商標または登録商標です。

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。