Oracle® Database

2 日で .NET 開発者ガイド

11*g* リリース 1（11.1）

部品番号 : E05695-02

2009 年 5 月

ORACLE®

Oracle Database 2 日で .NET 開発者ガイド , 11*g* リリース 1（11.1）

部品番号 : E05695-02

Oracle Database 2 Day + .NET Developer's Guide, 11*g* Release 1 (11.1)

原本部品番号 : B28844-02

原本著者 : Janis Greenberg、Roza Leyderman

原本協力者 : John Paul Cook、Mark Williams、Alex Keh、Christian Shay

目次

3 ODP.NET による単純な .NET アプリケーションの作成

4 Oracle Data Provider for .NET での取得と更新

5 Oracle Developer Tools for Visual Studio の使用

6 PL/SQL ストアド・プロシージャおよび REF CURSOR の使用

7 Oracle Database での ASP.NET の使用

8 .NET ストアド・プロシージャの開発とデプロイ

9 グローバリゼーション・サポートの組込み

A Oracle Database インスタンスの起動および停止

B フォームのコピー

索引

はじめに

このマニュアルでは、Microsoft .NET Framework に対応する Oracle テクノロジを使用した、Oracle Database でのアプリケーション開発について説明します。

対象読者

このマニュアルは、『Oracle Database 2 日でデータベース管理者』および『Oracle Database 2 日で開発者ガイド』をすでに読んでいるユーザーで、SQL および PL/SQL の基礎知識があり、Microsoft Visual Studio の使用方法を理解しているユーザーを対象としています。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様を含む、すべてのお客様にオラクル社の製品、サービスおよびサポート・ドキュメントをご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト http://www.oracle.com/accessibility/ を参照してください。

ドキュメント内のサンプル・コードのアクセシビリティについて

スクリーン・リーダーは、ドキュメント内のサンプル・コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかし JAWS は括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

Oracle サポート・サービスへの TTY アクセス

AT&T カスタマ・アシスタントに連絡するには、711 または +1-800-855-2880 にお電話ください。AT&T カスタマ・アシスタントによるお客様と Oracle サポート・サービス間の情報のリレーについては、+1-800-223-1711 にお電話ください。AT&T リレー・サービスの使用方法の詳細は、`http://www.consumer.att.com/relay/tty/standard2.html` を参照してください。AT&T カスタマ・アシスタントが Oracle サポート・サービスに連絡した後、Oracle サポート・サービスのエンジニアは技術的な問題に対処し、Oracle サービス・リクエスト・プロセスに従ってお客様をサポートします。

関連ドキュメント

詳細は、次の Oracle Database ドキュメントを参照してください。

- 『Oracle Data Provider for .NET 開発者ガイド』
- 『Oracle Database Extensions for .NET 開発者ガイド』
- 『Oracle Database 2 日でデータベース管理者』
- 『Oracle Database 2 日で開発者ガイド』
- Oracle Developer Tools for Visual Studio のダイナミック・ヘルプ
- 『Oracle Database Net Services 管理者ガイド』
- 『Oracle Database Express Edition Installation Guide for Microsoft Windows』

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

サポートおよびサービス

次の各項に、各サービスに接続するための URL を記載します。

Oracle サポート・サービス

オラクル製品サポートの購入方法、および Oracle サポート・サービスへの連絡方法の詳細は、次の URL を参照してください。

```
http://www.oracle.com/lang/jp/support/index.html
```

製品マニュアル

製品のマニュアルは、次の URL にあります。

```
http://www.oracle.com/technology/global/jp/documentation/index.html
```

研修およびトレーニング

研修に関する情報とスケジュールは、次の URL で入手できます。

```
http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=3
```

その他の情報

オラクル製品やサービスに関するその他の情報については、次の URL から参照してください。

```
http://www.oracle.com/lang/jp/index.html
http://www.oracle.com/technology/global/jp/index.html
```

注意： ドキュメント内に記載されている URL や参照ドキュメントには、Oracle Corporation が提供する英語の情報も含まれています。日本語版の情報については、前述の URL を参照してください。

1 概要

この章の内容は次のとおりです。

このマニュアルについて

このマニュアルは、クイック・スタート・ガイドであり、Oracle Data Provider for .NET および Oracle Developer Tools for Visual Studio の主な機能をはじめとする、Microsoft .NET Framework に対応した Oracle テクノロジについて説明しています。インストールと構成、Oracle .NET 製品を使用した基本的なアプリケーションの作成方法、PL/SQL および .NET ストアド・プロシージャの両方の作成方法と使用方法についても説明します。

注意： このマニュアルは Microsoft Visual Studio 2008 を使用して作成されました。Microsoft Visual Studio 2005 をご使用の場合は、スクリーン・ショット、ショートカット、メニュー・オプションおよび生成されるコードが若干異なることはありますが、通常、問題が発生することはありません。

このマニュアルの内容を理解した後は、Oracle Database JP Documentation Library に含まれる様々な情報の理解に進むことができます。

参照：

- Visual Studio のダイナミック・ヘルプ
- 『Oracle Data Provider for .NET 開発者ガイド』
- 『Oracle Database Extensions for .NET 開発者ガイド』
- 『Oracle Database 2 日でデータベース管理者』
- 『Oracle Database 2 日で開発者ガイド』

Microsoft .NET Framework の概要

Microsoft .NET Framework は、アプリケーションや XML Web サービスを作成、デプロイおよび実行するための多言語環境です。主なコンポーネントは次のとおりです。

共通言語ランタイム

共通言語ランタイム（CLR）は、実行中のアプリケーションの管理に役立つサービスを提供する、言語に依存しない開発環境およびランタイム環境です。

Framework クラス・ライブラリ

Framework クラス・ライブラリ（FCL）には、事前にパッケージ化された機能における一貫性のあるオブジェクト指向ライブラリが含まれています。

Oracle Data Provider for .NET の概要

Oracle Data Provider for .NET（ODP.NET）は、.NET クライアント・アプリケーションから Oracle Database への高速かつ効率的な ADO.NET データ・アクセス、およびその他の Oracle Database 機能へのアクセスを提供します。

開発者は、ODP.NET を使用して、Real Application Clusters や XML DB などの Oracle Database の拡張機能、および高度なセキュリティを利用できます。

Oracle Developer Tools for Visual Studio の概要

Oracle Developer Tools for Visual Studio（ODT）は、Visual Studio 環境と統合された一連のアプリケーション・ツールです。これらのツールでは、Oracle 機能に対し、Graphical User Interface によってアクセスを行います。また、ユーザーは様々なアプリケーション開発タスクを実行できる他、開発の生産性や使いやすさが向上します。Oracle Developer Tools では、Visual Basic、C#、およびその他の .NET 言語による .NET ストアド・プロシージャのプログラミングおよび実装をサポートしています。

次に、Oracle Developer Tools の機能の一部を示します。

- Oracle スキーマを参照するための Server Explorer との統合
- スキーマ・オブジェクトを作成および変更するためのデザイナおよびウィザード
- コードを自動生成するためにスキーマ・オブジェクトを .NET フォーム上にドラッグ・アンド・ドロップする機能
- 状況依存のダイナミック・ヘルプが統合された PL/SQL エディタおよびデバッガ
- データの挿入および更新、Visual Studio 環境でのストアド・プロシージャのテストなど、データベースの日常的なタスクを実行するための Oracle Data Window
- SQL 文または PL/SQL スクリプトを実行するための Oracle Query Window
- Oracle Deployment Wizard for .NET（1-3 ページの「Microsoft Visual Studio との統合」を参照）

.NET ストアド・プロシージャの概要

Oracle Database Extensions for .NET は、Windows での Oracle Database のデータベース・オプションです。このオプションによって、Microsoft Windows 用の Oracle Database を使用する .NET ストアド・プロシージャまたはファンクションを、Visual Basic .NET または Visual C# を使用して作成および実行できるようになります。

参照：『Oracle Database Extensions for .NET 開発者ガイド』

Microsoft Visual Studio との統合

.NET アセンブリに .NET プロシージャおよびファンクションを作成した後は、Oracle Developer Tools for Visual Studio のコンポーネントである Oracle Deployment Wizard for .NET を使用してそれらを Oracle Database にデプロイできます。

Oracle Providers for ASP.NET の概要

ASP.NET の開発者は、Oracle Providers for ASP.NET を使用すると、Web アプリケーション（Web ユーザー情報、ショッピング・カートなど）に共通のアプリケーション状態を Oracle Database に簡単に格納できるようになります。これらのプロバイダは、既存の Microsoft ASP.NET プロバイダをモデルにしており、類似スキーマおよびプログラミング・インタフェースを共有することで、.NET 開発者に使い慣れたインタフェースを提供します。

Oracle では、次のプロバイダがサポートされています。

- メンバーシップ・プロバイダ
- ロール・プロバイダ
- サイト・マップ・プロバイダ
- セッション状態プロバイダ
- プロファイル・プロバイダ
- Web イベント・プロバイダ
- Web パーツ・パーソナライズ・プロバイダ
- キャッシュ依存性プロバイダ

各 ASP.NET プロバイダは、Web サイトで個々に使用するか、または他の Oracle ASP.NET プロバイダと組み合せて使用できます。それぞれに、特定の Web サイト情報が格納されます。

Oracle Providers for ASP.NET のクラス、その使用方法、インストールおよび要件については、『Oracle Providers for ASP.NET 開発者ガイド』を参照してください。この内容は、ダイナミック・ヘルプとしても提供されています。

参照：

- 第 7 章「Oracle Database での ASP.NET の使用」
- 『Oracle Providers for ASP.NET 開発者ガイド』

2

.NET 製品のインストール

この章の内容は次のとおりです。

必要事項

この項では、このマニュアルで紹介する例を実行するために必要な製品とデータベース・スキーマを示します。

Oracle Database

ローカルまたはリモート・コンピュータのいずれかに、Oracle Database をインストールしておく必要があります。

> **注意：** このマニュアルで使用するすべての例で、Oracle Database 11g クライアントが必要です。ただし、このクライアントでは Oracle Database 9*i* リリース 2 以上がサポートされているため、このうちのいずれかのリリースを使用することもできます。
>
> Oracle Database Extensions for .NET を使用する予定がある場合は、クライアントを Oracle Database 11*g* に接続する必要もあります。

データベースは、ユーザー・インタフェースである Enterprise Manager を使用して管理できます。Enterprise Manager では、スクリプトや問合せを実行したりその他の操作を実行できます。

> **参照：** Oracle Database のインストールと構成が完了していない場合は、『Oracle Database Express Edition Installation Guide for Microsoft Windows』を参照してください。

サンプル・データ

このマニュアルで使用するサンプル・データは、Oracle サンプル・スキーマの 1 つである `HR` スキーマに含まれています。このサンプル・スキーマは、Oracle Database インストールの一部として含まれています。

> **参照：** HR のデータ・モデルおよび表については、『Oracle Database サンプル・スキーマ』を参照してください。

Oracle Data Access Components

Oracle Data Access Components（ODAC）は、次のツールのコレクションです。

- Oracle Developer Tools for Visual Studio
- Oracle Data Provider for .NET
- Oracle Providers for ASP.NET
- Oracle Provider for OLE DB
- Oracle Objects for OLE
- Oracle ODBC Driver
- Oracle Services for Microsoft Transaction Server
- Oracle SQL*Plus
- Oracle Instant Client

Oracle Database Extensions for .NET

Oracle Database Extensions for .NET は、Oracle Database 11*g* を Windows にインストールするときに、その一部としてインストールされます。Oracle Database Extensions for .NET をインストールすると、ODAC インストールによって、Oracle Database Extensions for .NET へのアップグレードが提供されます。このアップグレードは、Oracle Data Access Components for Oracle Server オプションの一部として含まれています（2-3 ページの「.NET 製品のインストール」の手順 4 のスクリーンショットを参照）。このマニュアルの第 8 章を完了する予定がある場合にのみ、このアップグレードを実行し、Oracle Database Extensions for .NET をインストールする必要があります。

Visual Studio のバージョン

Visual Studio 2008 を使用する場合は、インストールしてから、このマニュアルに示す手順に進む必要があります。

Microsoft Visual Studio 2005 を使用する場合は、スクリーン・ショット、ショートカット、メニュー・オプションおよび生成されるコードが若干異なることはありますが、通常、問題が発生することはありません。

.NET 製品のインストール

次の手順は、Visual Studio をインストールした後で行う、Oracle Developer Tools for Visual Studio（ODT）、Oracle Data Provider for .NET およびその他の ODAC 製品のインストール方法を示しています。

注意： 新しいバージョンの Oracle .NET 製品のリリースにより、このマニュアルに示しているインストール手順が若干変更される場合があることに注意してください。スクリーンショットは、Oracle Data Access Components（ODAC）バージョン 11.1.0.6.21 に基づいています。

インストールを行うには、次の手順を実行します。

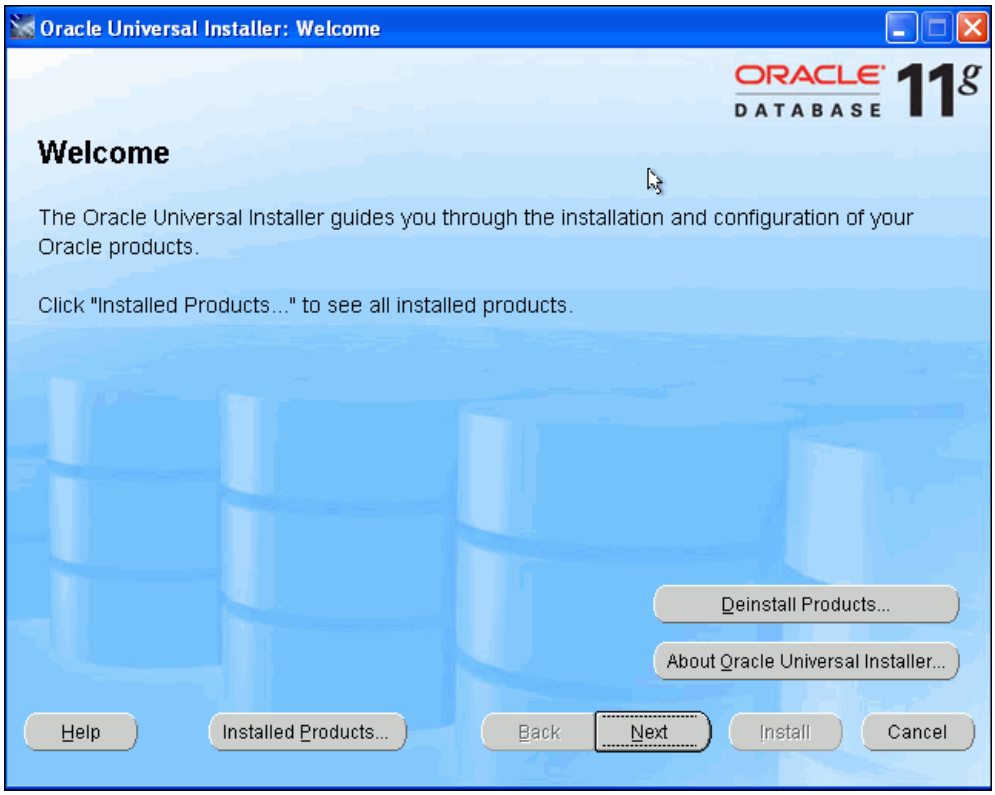
1. インターネット・ブラウザで、次の場所に移動して ODAC と Oracle Developer Tools for Visual Studio をダウンロードします。

   `http://www.oracle.com/technology/software/tech/windows/odpnet/index.html`

2. すべてのファイルを zip ファイルからファイル・システムのフォルダに抽出します。

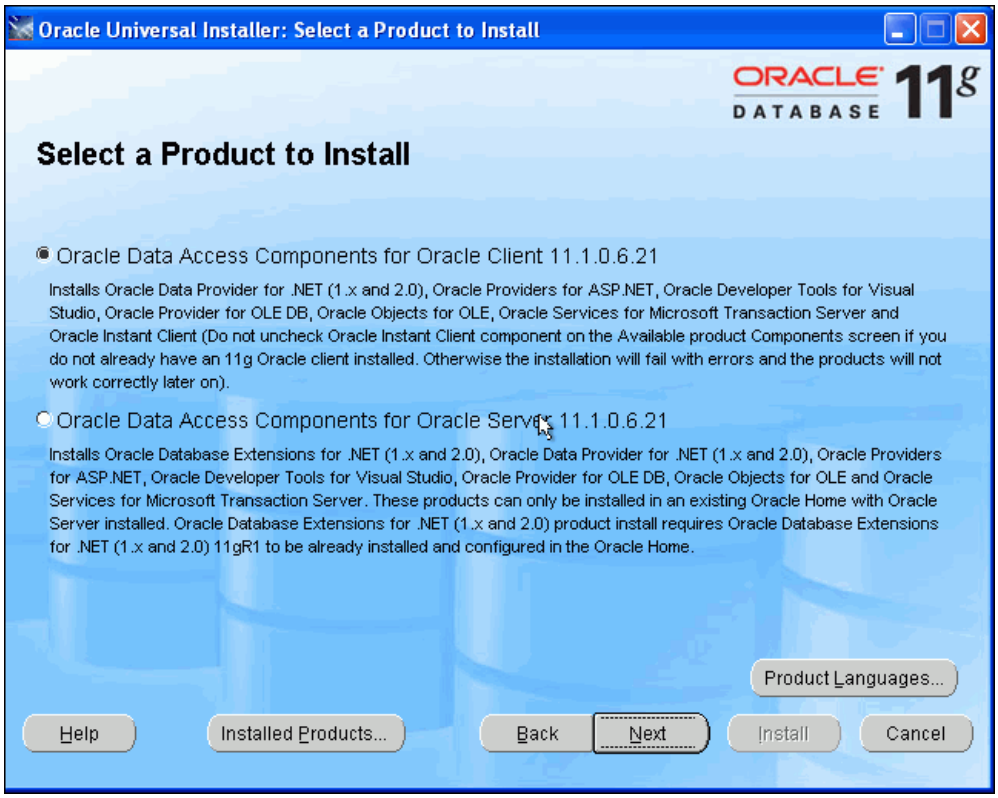
3. **Setup.exe** をダブルクリックします。

   Oracle Installer が起動します。必要な依存オブジェクトを検出する画面が短い間表示された後、Oracle Universal Installer（OUI）の「ようこそ」画面が表示されます。



4. **「次へ」**をクリックします。

   「インストールする製品の選択」画面が表示されます。

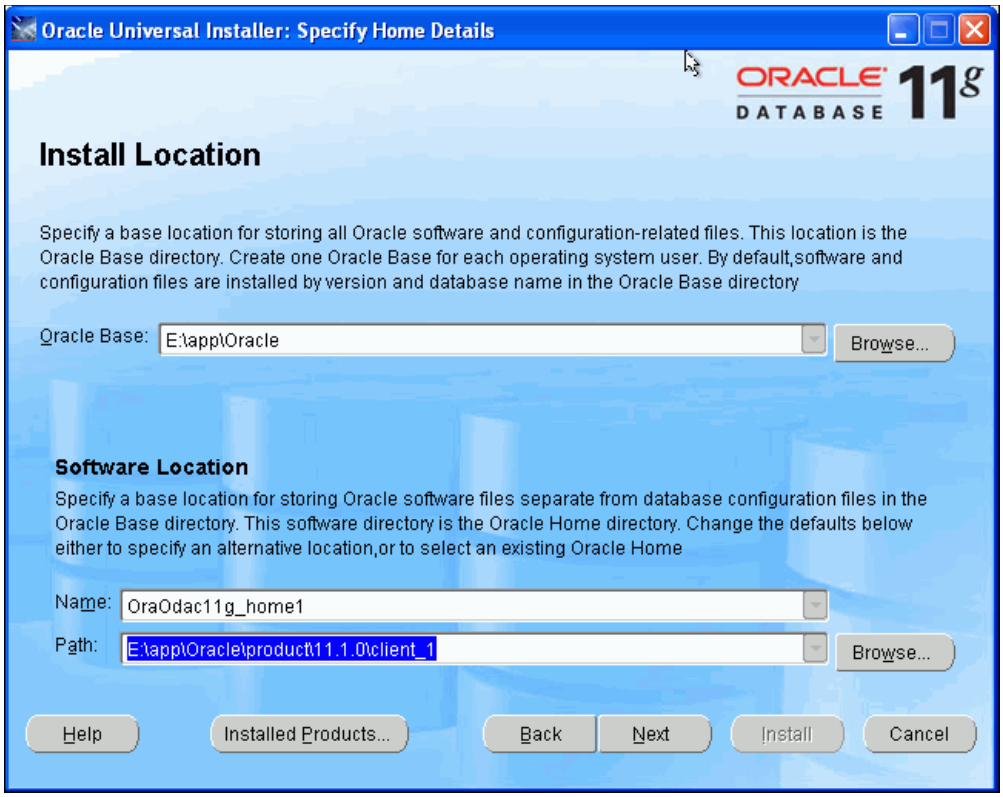


5. 最初のオプションを選択します。

   このオプション（ODAC for Oracle Client）を選択すると、クライアントの Oracle ホームで使用する製品のみがインストールされます。2 番目のオプション（ODAC for Oracle Server）を選択すると、Oracle Database が含まれる Oracle ホームに直接インストールできます。

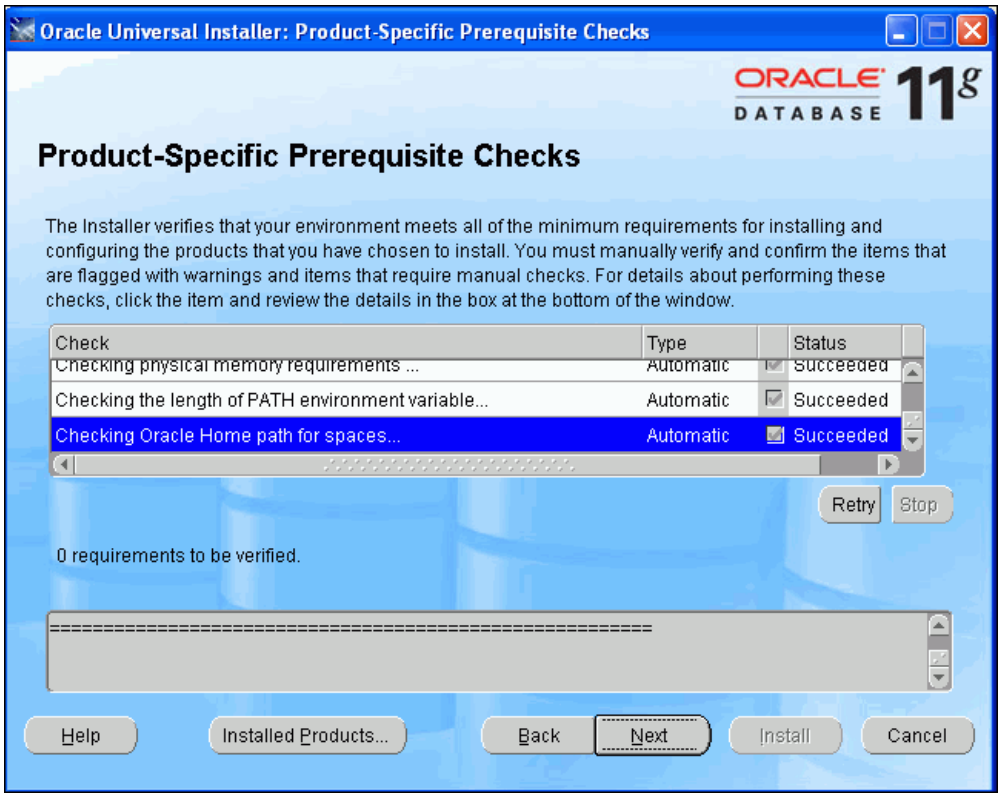
6. 「次へ」をクリックします。

「インストールの場所」ウィンドウが表示されます。このウィンドウで、インストールする場所を選択できます。デフォルトでは、クライアントの Oracle ホームが新しく作成されます。このマニュアルでは、デフォルトを受け入れて新しい Oracle ホームを作成します。



7. **「次へ」**をクリックします。

   インストーラにより前提条件のチェックが実行されます。各チェックのステータスは「成功しました」になる必要があります。

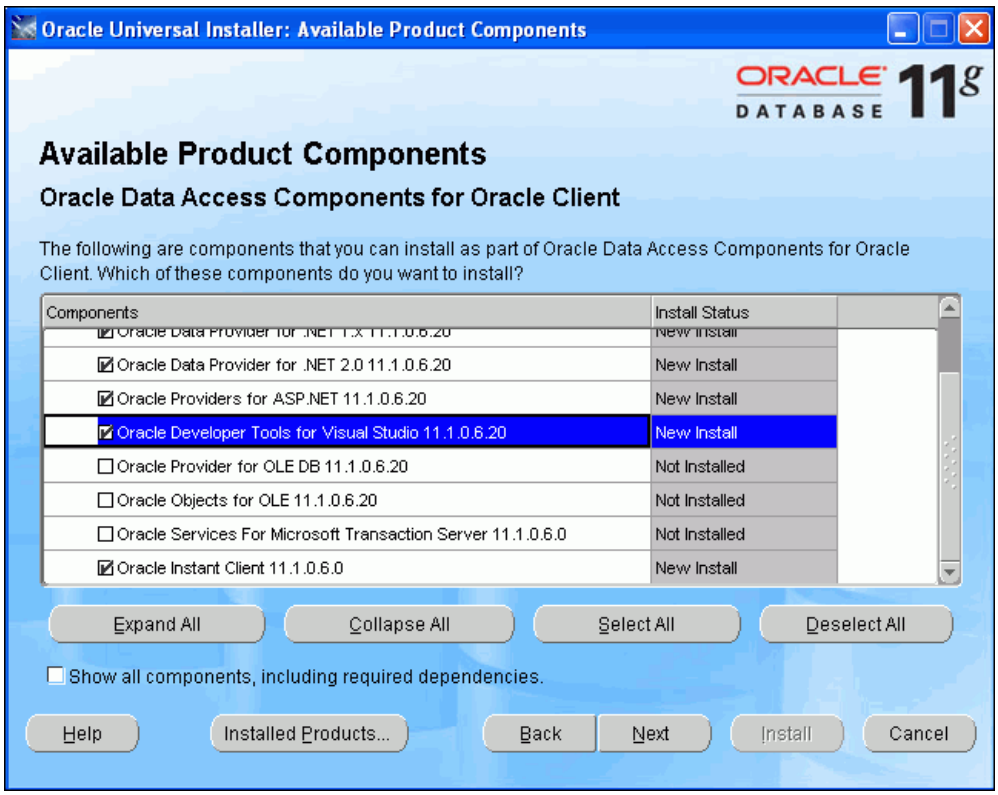


8. 「**次へ**」をクリックします。

   「使用可能な製品コンポーネント」画面が表示されます。

   次の製品が選択されていることを確認してください。

   - Oracle Data Provider for .NET 2.0
   - Oracle Providers for ASP.NET
   - Oracle Developer Tools for Visual Studio
   - Oracle Instant Client



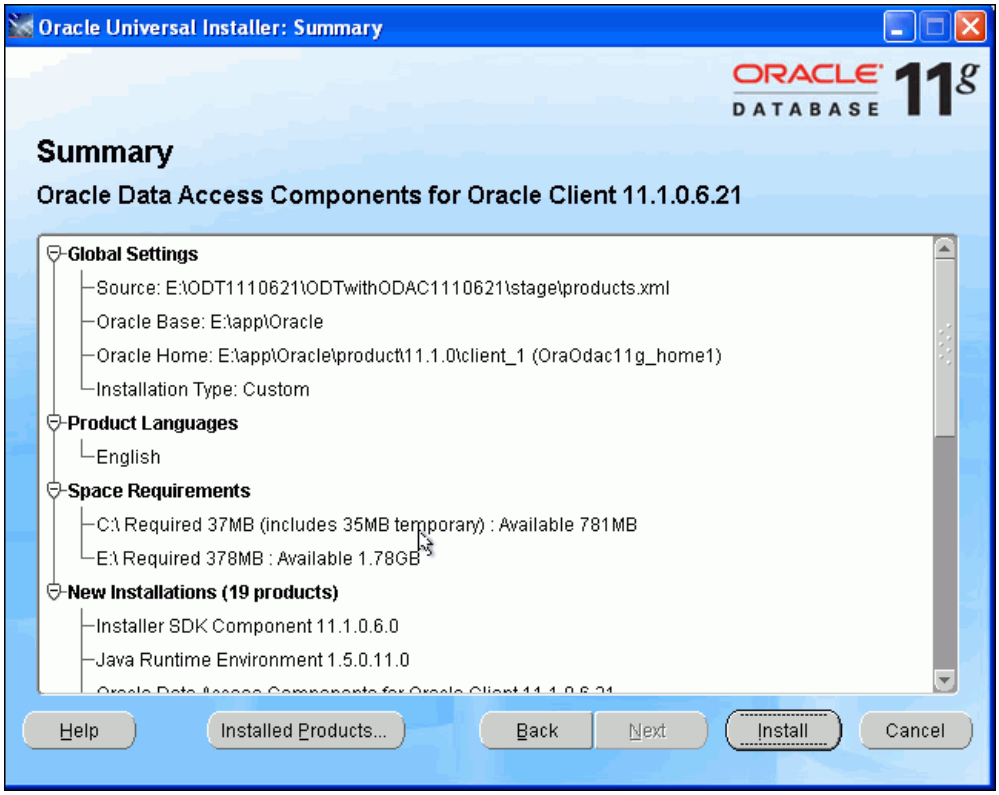
9. **「次へ」**をクリックします。

   Oracle Providers for ASP.NET を使用する場合は、*ORACLE_BASE*¥*ORACLE_HOME*¥client_1¥ASP.NET¥SQL にある SQL スクリプトを実行する必要があることを示す画面が表示されます。

   注意： *ORACLE_BASE*¥*ORACLE_HOME* は、Oracle ホームを表すディレクトリです。

10. **「次へ」**をクリックします。

    「サマリー」ウィンドウが表示されます。



11. **「インストール」**をクリックしてインストールを完了します。

    インストールの最後の画面が表示されます。再度、ASP.NET スクリプトをインストールするように指示されます。Oracle Providers for ASP.NET を使用する予定がある場合は、スクリプトをインストールします。

12. **「終了」**をクリックします。

NET 接続の別名の構成

Oracle クライアントが短縮名を使用してデータベースに接続できるように、tnsnames.ora ファイルにデータベース・サーバーのアドレスを定義します。DBA から事前構成済の tnsnames.ora ファイルがすでに提供されている場合もあります。

それ以外の場合は、*ORACLE_BASE*¥*ORACLE_HOME*¥network¥admin¥sample ディレクトリに移動して、そこにある tnsnames.ora ファイルと sqlnet.ora ファイルを *ORACLE_BASE*¥*ORACLE_HOME*¥network¥admin ディレクトリにコピーする必要があります。

tnsnames.ora ファイルに含まれる次の接続記述子を使用し、イタリック体で示されている値を使用環境に合わせて変更します。

例 2-1 tnsnames.ora 接続記述子

```
address name =
 (DESCRIPTION =
   (ADDRESS_LIST =
     (ADDRESS = (PROTOCOL = TCP)(Host = hostname)(Port = port))
   )
 (CONNECT_DATA =
   (SERVICE_NAME = sid)
 )
)
```

値の説明 :

sid: データベース・サービス名

hostname: データベース・コンピュータ名

port: データベースとの通信に使用するポート

address name: 接続記述子として使用するユーザー定義の短縮名。この短縮名は、.NET アプリケーションの接続文字列で使用します。

例 2-2 に、tnsnames.ora ファイルの例を示します。

例 2-2 tnsnames.ora ファイルの例

```
ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL)
    )
  )
```

参照 : 『Oracle Database Net Services 管理者ガイド』

Oracle Providers for ASP.NET の設定

Oracle Providers for ASP.NET を使用すると、データベース・ユーザーのスキーマのコンテキストに基づいて Oracle Database 内に Web アプリケーションの状態を格納できます。管理者は、新しいデータベース・ユーザーを作成してアプリケーションの状態を格納できます。

このデータベース・ユーザーは、単一の物理ユーザーにはマップされませんが、すべての Web サイト・ユーザーの ASP.NET 情報を格納するためのリポジトリとして機能します。したがって、単一または複数の Web ユーザーのアプリケーションの状態をこの新しいデータベース・ユーザーのスキーマ内に格納することができます。

注意： このチュートリアルでは、データベース・ユーザーは、ASPNET_DB_USER と呼ばれ、Oracle Providers for ASP.NET データベース・ユーザーです。

実行時に、ASP.NET アプリケーションは、接続文字列にデータベース・ユーザーの資格証明を使用してデータベースに接続します。

Oracle Database を設定するには、データベース管理者が Oracle Providers for ASP.NET データベース・ユーザーのスキーマに特定のデータベース権限を付与する必要があります。データベース・ユーザーは、これらの権限を使用して、表、ビュー、ストアド・プロシージャなどの Oracle Providers for ASP.NET で必要なデータベース・オブジェクトを作成できます。

権限が付与されると、データベース・ユーザーは、Oracle Provider for ASP.NET 構成スクリプトを実行できます。

参照： 詳細は、『Oracle Providers for ASP.NET 開発者ガイド』を参照してください。

この項では、データベースに Oracle Providers for ASP.NET を設定する手順について説明します。このチュートリアルの ASP.NET プロバイダに関する項（第 7 章の後半）を学習する予定がない場合は、この項も学習する必要はありません。Oracle Providers for ASP.NET の設定では、Oracle Developer Tools for Visual Studio を使用するため、設定を開始する前にインストールしておく必要があります。

この項の内容は次のとおりです。

- Oracle Providers for ASP.NET データベース・ユーザーの設定
- 接続文字列の設定
- 様々な設定に対する Oracle Providers for ASP.NET のカスタマイズ

Oracle Providers for ASP.NET データベース・ユーザーの設定

このチュートリアルでは、新しいデータベース・ユーザー・スキーマ ASPNET_DB_USER を作成して ASP.NET プロバイダのデータを格納します。ユーザー ASPNET_DB_USER に特定のデータベース権限を付与し、ASP.NET プロバイダのデータベース・スクリプトを実行してスキーマを設定します。このスキーマに、Oracle Providers for ASP.NET に必要な表、ストアド・プロシージャなどのデータベース・オブジェクトが格納されます。

この項の内容は次のとおりです。

- ユーザーの作成および権限の付与
- Oracle Providers for ASP.NET のすべての構成
- Oracle Providers for ASP.NET の個別構成
- Oracle Providers for ASP.NET に対するスキーマのアンインストール

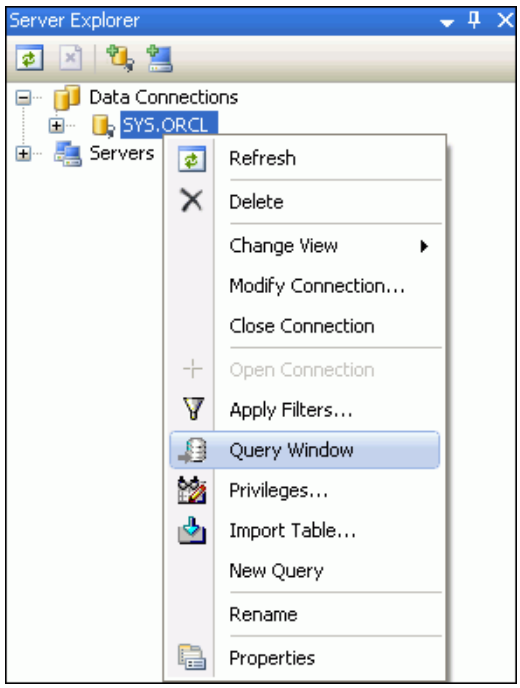
ユーザーの作成および権限の付与

新しいユーザーを追加し、必要な権限を付与するには、次の手順を実行します。

1. SYS または別のデータベース管理ユーザーとしてログインします。詳細は、8-3 ページの「SYSDBA としての接続の作成」を参照してください。

2. Server Explorer の問合せウィンドウで、新しい ASPNET_DB_USER ユーザーを次のように作成します。

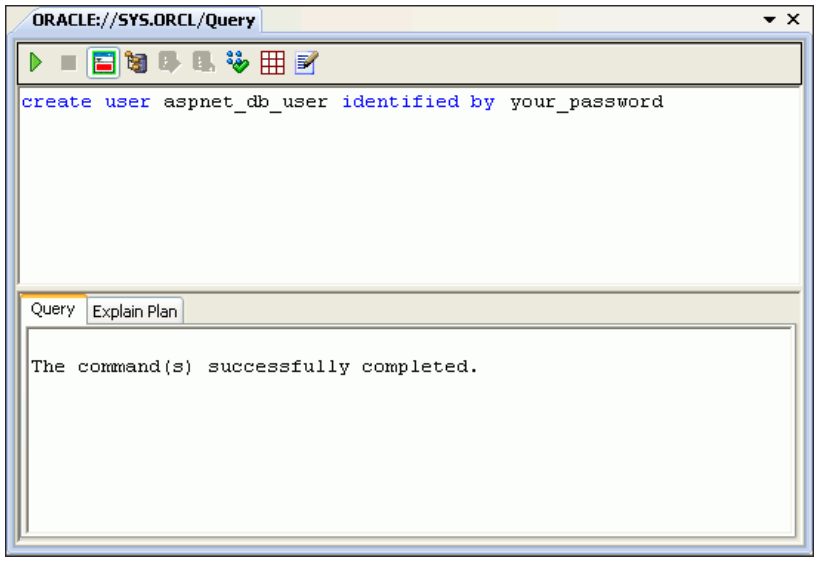
   a. Server Explorer で、SYS.ORCL を右クリックし、「問合せウィンドウ」を右クリックします。



   b. 問合せウィンドウで、次のコマンドを入力します。

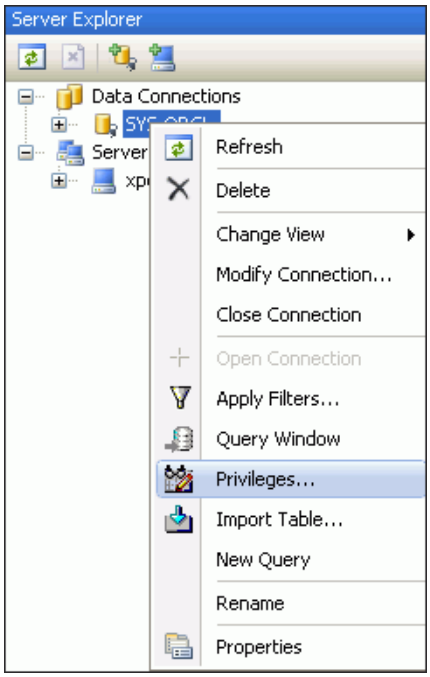
   ```
   create user ASPNET_DB_USER identified by your_password
   ```

   この手順によって、入力したパスワードを持つユーザー ASPNET_DB_USER がデータベースに作成されます。



c. 「問合せの実行」（左上の緑色の矢印）をクリックしてコマンドを実行します。下部のウィンドウに、コマンドが正常に実行されたことが示されます。

3. Server Explorer に戻り、SYS.ORCL を再度選択して右クリックし、**「権限」**を選択します。



これによって、ODT の「権限の付与 / 取消し」ウィザードが起動されます。

4. スキーマを作成し、ASP.NET プロバイダの Web サイトの状態を格納できるように新しいデータベース・ユーザーに権限を付与します。

- 「オブジェクト・タイプ」を USER、「ユーザー」を ASPNET_DB_USER に設定します。

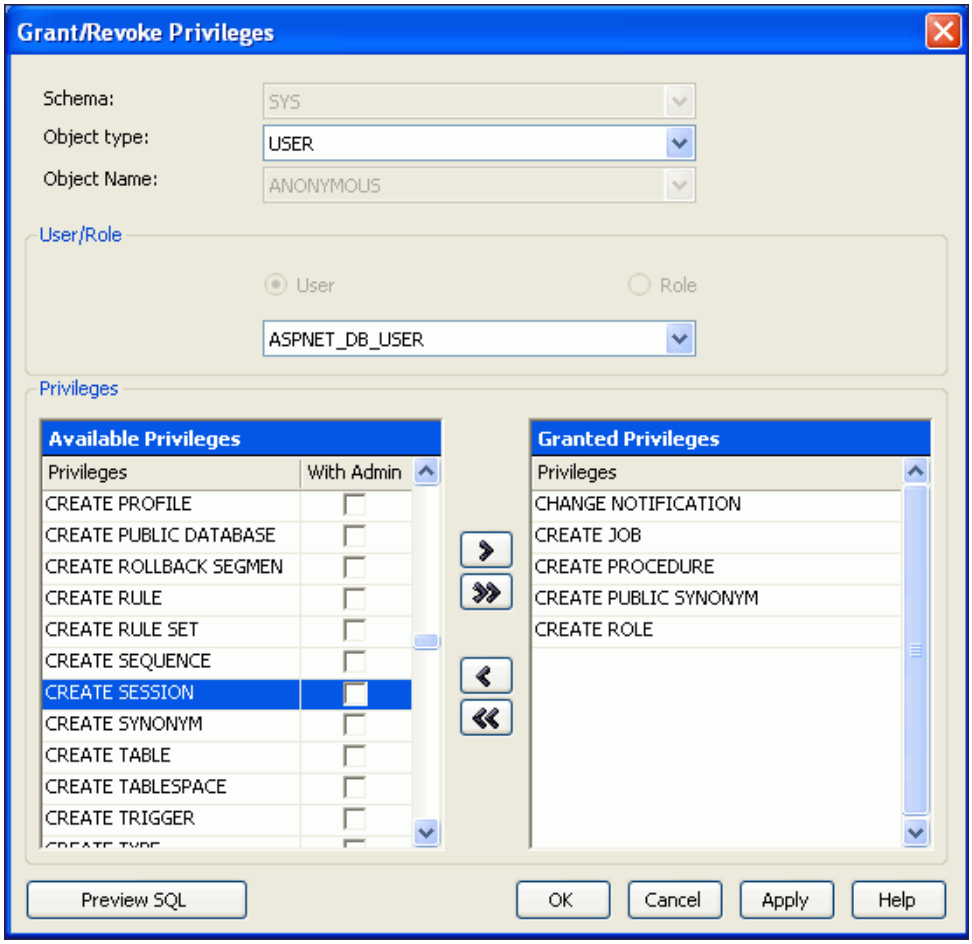
注意： ASPNET_DB_USER を表示するには、リフレッシュが必要な場合があります。

- 中央にある右向き矢印（>）を使用して、権限を「使用可能な権限」リストから「付与された権限」リストに移動します。

  通常必要な権限は次のとおりです。

  - 変更の通知
  - ジョブの作成
  - プロシージャの作成
  - パブリック・シノニムの作成
  - ロールの作成
  - セッションの作成
  - 表の作成
  - ビューの作成

- パブリック・シノニムの削除
- 無制限表領域：この例では、ASPNET_DB_USER 無制限表領域が付与されます。ただし、ほとんどの場合、管理者がデータベース・ユーザーに特定の表領域割当て制限を割り当てます。



「**適用**」をクリックすると、出力ウィンドウに正常に実行されたことが示されます。「**OK**」をクリックします。

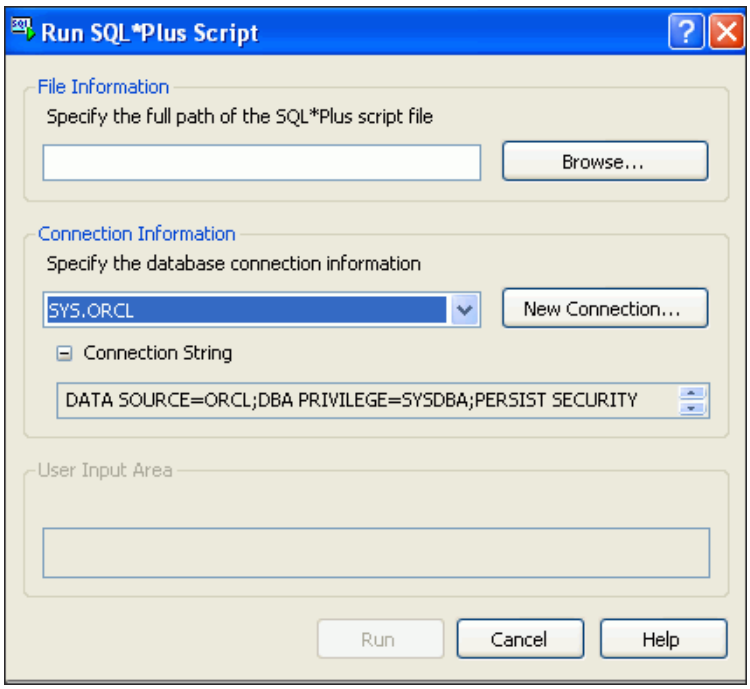
Oracle Providers for ASP.NET ユーザーに必要な権限が付与されていない場合は、設定スクリプトの実行時にエラーが発生することがあります。

Oracle Providers for ASP.NET のすべての構成

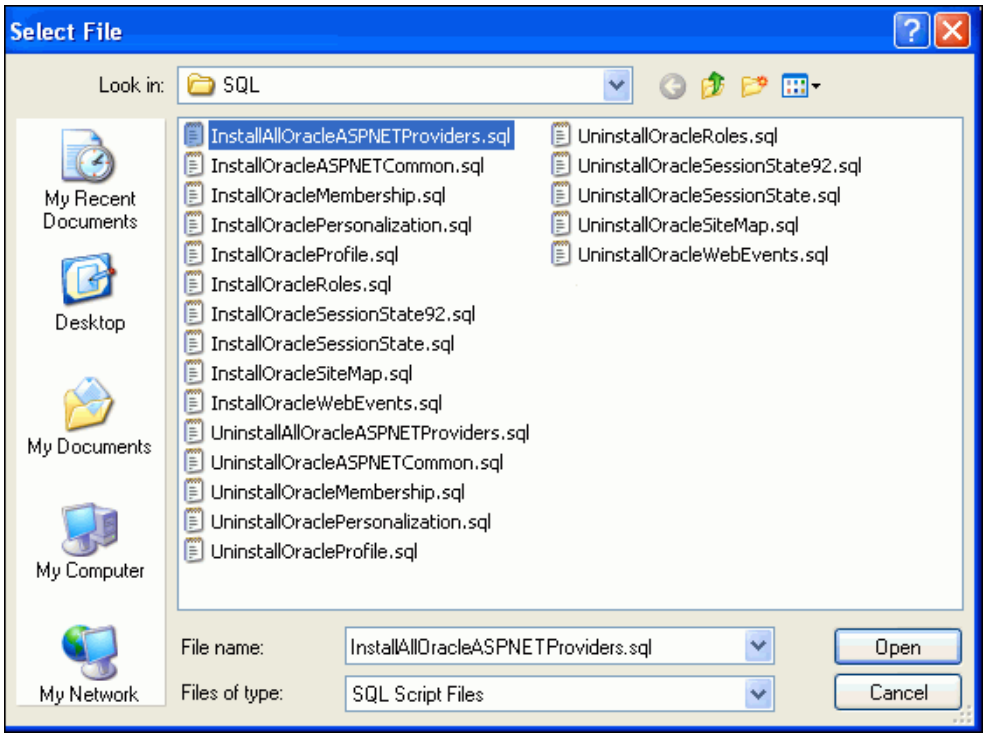
データベースですべてのプロバイダをすぐに構成するには、`InstallAllOracleASPNETProviders.sql` を実行します。

Oracle Developer Tools でこのスクリプトを実行するには、次の手順を実行します。

1. Visual Studio で、**「ツール」**を選択してから**「SQL*Plus スクリプトの実行」**を選択します。画面が表示されたら、**「参照」**を選択します。

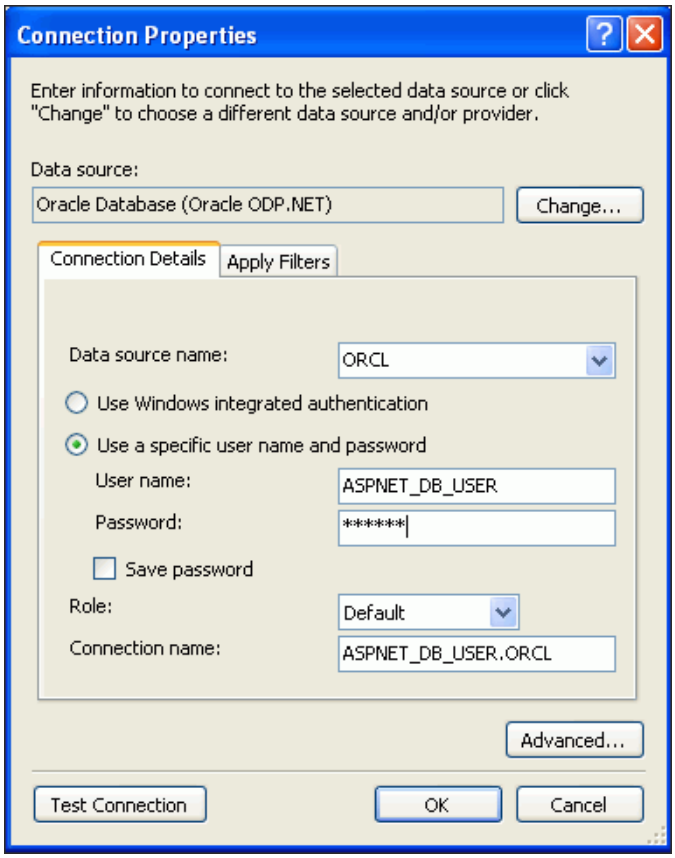


2. *ORACLE_BASE*¥*ORACLE_HOME*¥ASP.NET¥sql ディレクトリ（*ORACLE_BASE*¥*ORACLE_HOME* は Oracle ホームを表す）を参照し、InstallAllOracleASPNETProviders.sql を選択して「**開く**」を選択します。



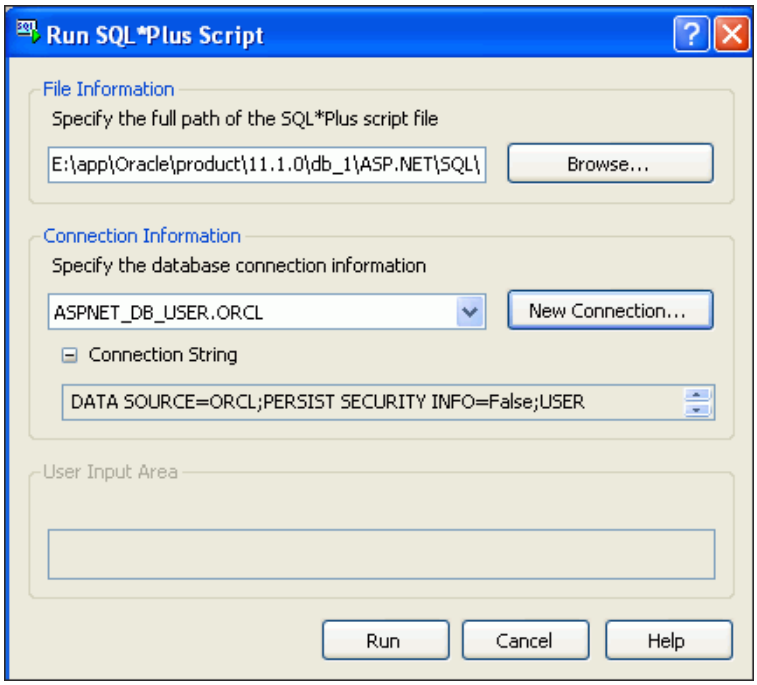
3. **「SQL*Plus スクリプトの実行」**画面が再度表示されたら、**「新規接続」**を選択します。

「接続プロパティ」画面が表示されたら、データソースが Oracle Database（Oracle ODP.NET）であり、データソース名が ORCL であることを確認します。次に、「ロール」を「デフォルト」に設定して、ユーザー名 ASPNET_DB_USER およびパスワードを入力します。**「OK」**をクリックします。



「Oracle サーバー・ログイン」ダイアログ・ボックスが表示される場合もあります。その場合は、パスワードを保存するかどうかを選択できます。

4. 「SQL*Plus スクリプトの実行」が再度表示されたら、**「実行」**を選択します。



SQL ファイルが実行され、バックグラウンドでは、スクリプトが正常に実行されたことが出力ウィンドウで確認されます。

5. スクリプトの実行が終了したら、**「取消」**を選択します。

Oracle Providers for ASP.NET の個別構成

アプリケーションにすべての Oracle Providers for ASP.NET が必要ない場合もあります。プロバイダは、個別に設定することができます。通常は、他のインストール・スクリプトを実行する前に `InstallOracleASPNETCommon.sql` インストール・スクリプトを実行する必要があります。これによって、ASP.NET プロバイダに共通のインフラストラクチャが設定されます。その後、必要な Oracle Provider for ASP.NET に対して個別に特定の SQL スクリプトを（任意の順序で）実行します。

これらのインストール・スクリプトは、*ORACLE_BASE*¥*ORACLE_HOME*¥ASP.NET¥sql ディレクトリにあります。

表 2-1 Oracle Providers for ASP.NET の個別のインストール・スクリプト

プロバイダ	必要なインストール・スクリプト
Oracle Membership Provider	`InstallOracleMembership.sql`
Oracle Personalization Provider	`InstallOraclePersonalization.sql`
Oracle Profile Provider	`InstallOracleProfile.sql`
Oracle Role Provider	`InstallOracleRoles.sql`

表 2-1 Oracle Providers for ASP.NET の個別のインストール・スクリプト（続き）

プロバイダ	必要なインストール・スクリプト
Oracle Session State Provider	`InstallOracleSessionState.sql` （Oracle Database 10*g* リリース 1 以上） `InstallOracleSessionState92.sql`（Oracle Database 9*i* リリース 2） これらのインストール・スクリプトには、対応する名前が付いたアンインストール・スクリプトがあります。 **注意：**このプロバイダでは、`InstallOracleASPNETCommon.sql` を実行する必要はありません。実行する必要があるのは、プロバイダ固有の `.sql` インストール・スクリプトのみです。
Oracle Site Map Provider	`InstallOracleSiteMap.sql`
Oracle Web Events Provider	`InstallOracleWebEvents.sql`
Oracle Cache Dependency Provider	スクリプトを実行する必要はありません。

Oracle Providers for ASP.NET に対するスキーマのアンインストール

対応するアンインストール・スクリプトを使用して、インストール・スクリプトで作成したデータベース・オブジェクトを削除します。これらのスクリプトには、接頭辞 `Uninstall` が付いています。

接続文字列の設定

Oracle Providers for ASP.NET の情報を格納および取得するようにデータベースが構成されたら、中間層またはクライアントを `ASPNET_DB_USER` ユーザーに接続する必要があります。

使用しているコンピュータに接続情報を構成するには、次の手順を実行します。

1. *drive*`:¥WINDOWS¥Microsoft.NET¥Framework¥v2.0.50727¥CONFIG` にある `machine.config` ファイルに移動します。
2. テキスト・エディタを使用して、`<connectionStrings>` を検索し、`<add name="OraAspNetConString"..` で始まる行を変更して、ユーザー ID、パスワード、データソース・エントリおよびプロバイダ名を次のように追加します。

```
<connectionStrings>
<add name="OraAspNetConString" connectionString="User
     Id=aspnet_db_user;Password=your_password;Data Source=orcl;"
     providerName="Oracle.DataAccess.Client" />
</connectionStrings>
```

> **注意：** パスワードは、事前に作成したパスワードに変更してください。また、接続文字列にコピーした可能性のあるキャリッジ・リターンは削除してください。

様々な設定に対する Oracle Providers for ASP.NET のカスタマイズ

開発者は、各 ASP.NET プロバイダのプロパティを machine.config ファイルの <system.web> セクション内からカスタマイズできます。

machine.config ファイルは Oracle Universal Installer によって自動的に構成されますが、開発者は、web.config ファイルを使用して、Oracle Providers for ASP.NET に対してより詳細なアプリケーション・レベルの制御を適用できます。このファイルによって、machine.config ファイルからのエントリは変更されますが、関連付けられている特定の Web アプリケーションの場合にかぎります。開発者は、machine.config ファイルと同じ XML 構文を使用して web.config ファイルを設定できます。

3

ODP.NET による単純な .NET アプリケーションの作成

この章の内容は次のとおりです。

新しいプロジェクトの作成

Visual Studio では、作成するすべての開発コードがグループ化され、プロジェクトと呼ばれるコンテナに入れられます。単純なプロジェクトのほとんどは、1 つのファイルのみで構成されます。この項では、新しい開発プロジェクトの作成方法を説明します。

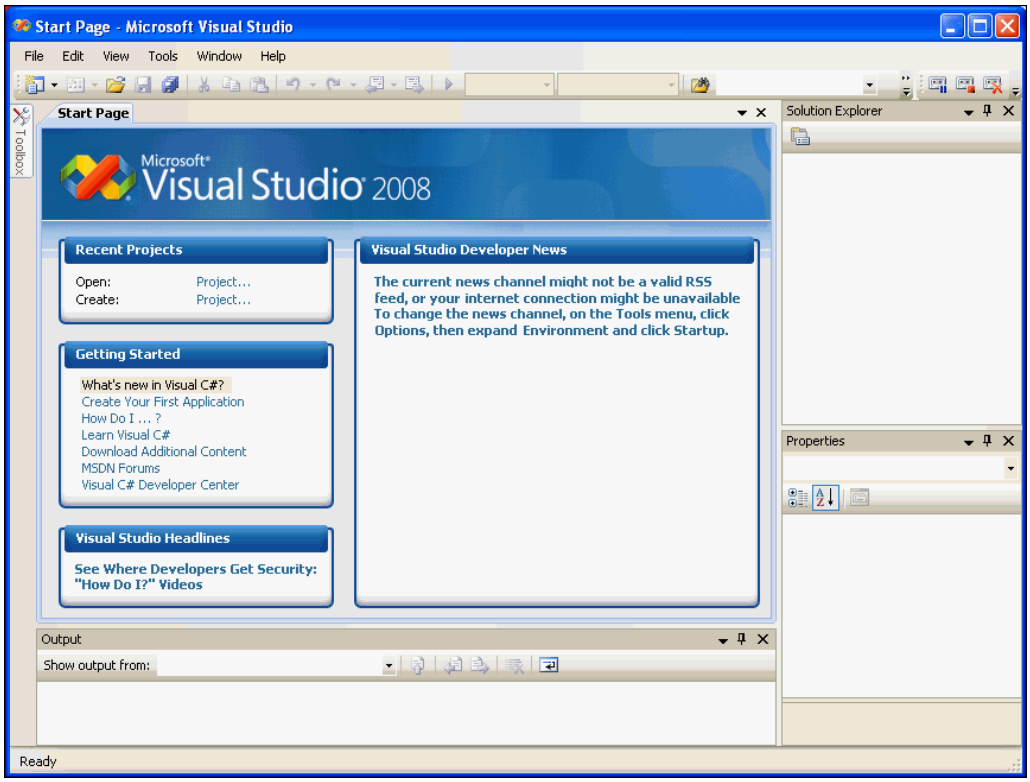
この章で作成するアプリケーションは、後続の章で行う操作の開始点になるため、このマニュアルに示す順に操作してください。

注意 : 必要に応じて、**Visual C#** または **Visual Basic** が指定されています。

新しいプロジェクトを開始するには、次の手順を実行します。

1. Visual Studio を起動します。

   「スタート」メニューを開き、**「すべてのプログラム」**から**「Microsoft Visual Studio 2008」**を選択します。



   Microsoft Visual Studio IDE 環境が表示されます。

2. 「Start Page」の「Recent Projects」ヘッダーの下にある**「Create: Project」**をクリックします。

   または、**「File」**メニューから**「New」**、**「Project」**の順に選択します。

   「New Project」ダイアログ・ボックスが表示されます。

3. 「Project Types」で、作成するプロジェクトの種類を選択します。

   Visual C#:

   Visual C#: Windows

   Visual Basic:

   Other Languages: Visual Basic: Windows

4. 「Templates」で**「Windows Forms Application」**を選択します。
5. 「Name」フィールドに適切な名前を入力します。

   Visual C#:

   HR_Connect_CS

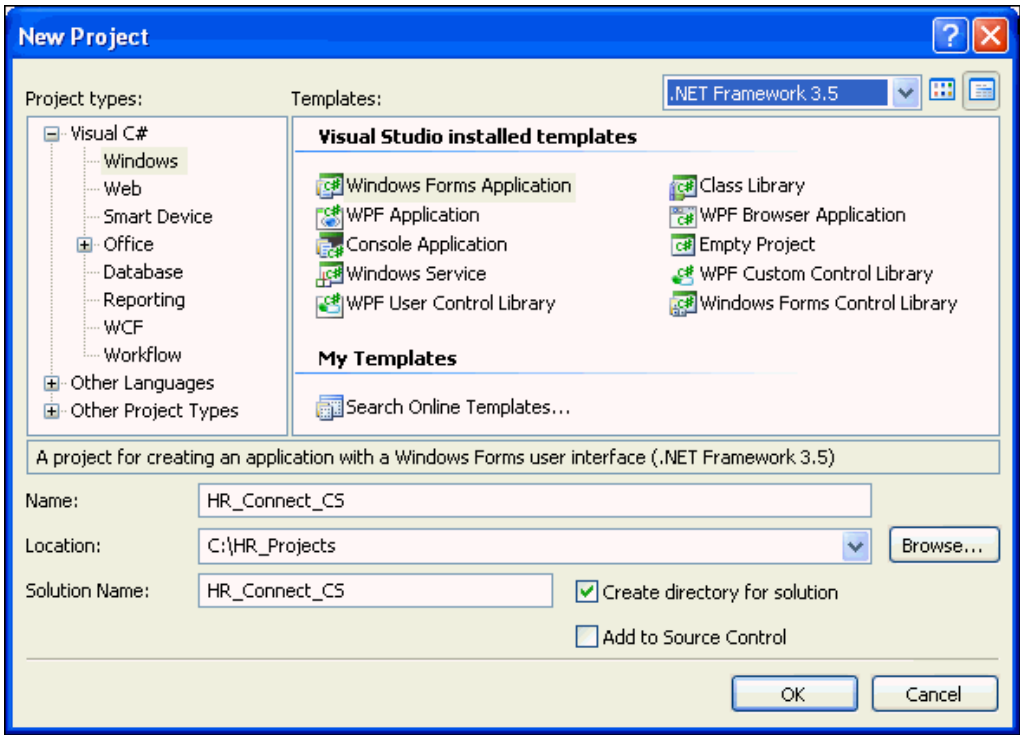
   Visual Basic:

   HR_Connect_VB

   略語の CS は C# プロジェクトを、VB は Visual Basic プロジェクトを示します。
6. 「Location」にファイルを保存するディレクトリを入力します。

   このマニュアルに合わせて、このディレクトリを `C:¥HR_Projects` と入力します。
7. 「Solution Name」に適切な名前（HR_Connect_CS または HR_Connect_VB）が表示されます。

   1 つのソリューションに複数のプロジェクトを格納できます。ソリューションに含まれるプロジェクトが 1 つのみの場合は、両方に同じ名前を使用できます。
8. **「Create directory for solution」**を選択します。
9. **「OK」**をクリックします。

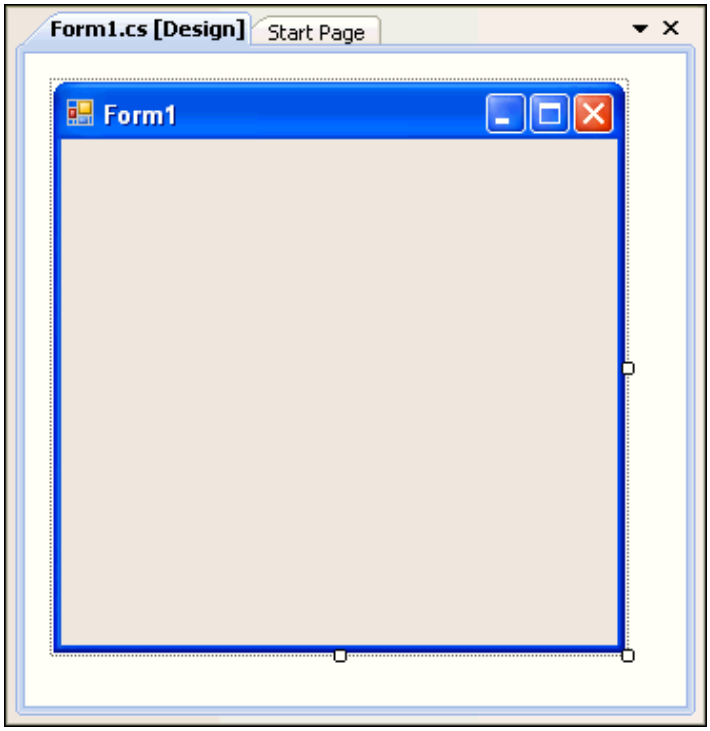


プロジェクトが作成されます。

メイン・ウィンドウには、言語に応じて「HR_Connect_CS - Microsoft Visual Studio」または「HR_Connect_VB - Microsoft Visual Studio」のいずれかが新しいタイトルとして表示され、次に示す Form1 が作成されます。

多くのプロジェクトの最初のフォームには、Form1 という名前が自動的に付けられることに注意してください。これはフォーム・コントロールの名前です。この名前と、コード・ファイルに付けられる実際の名前を混同しないでください。通常、コード・ファイルの名前は `Form1.cs` または `Form1.vb` です。

Form1 と *Form1.xx* は、両方とも名前を変更できます。このマニュアルでは、*Form1.xx* の名前を何回か変更します。

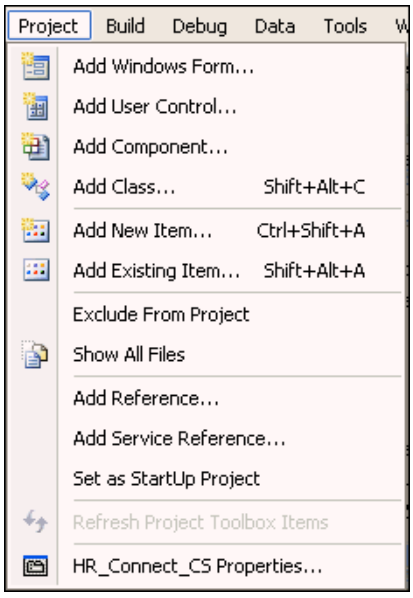


参照の追加

この項では、データ・プロバイダである Oracle Data Provider for .NET が含まれる `Oracle.DataAccess.dll` ファイルへの参照を追加する方法を説明します。

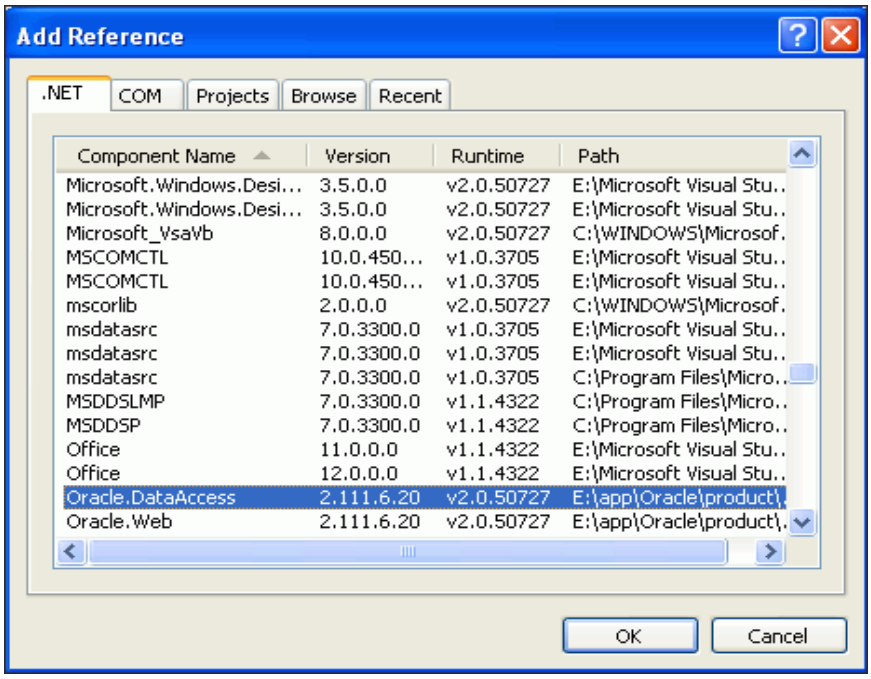
参照を追加するには、次の手順を実行します。

1. 「**Project**」メニューから「**Add Reference**」を選択します。

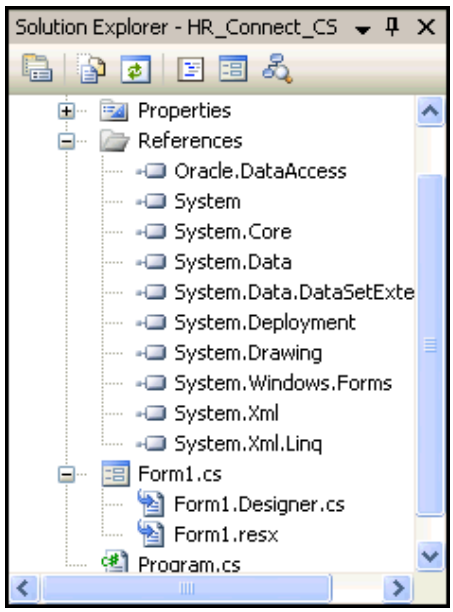


「Add Reference」ウィンドウが表示されます。

2. 「Add Reference」ウィンドウの「.NET」タブで **Oracle.DataAccess** を選択します。**「OK」**をクリックします。



「Solution Explorer」に新しい参照が表示されていることを確認します。



名前空間ディレクティブの追加

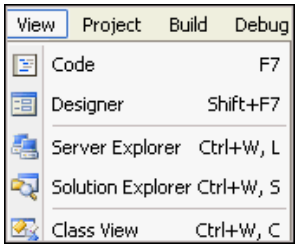
モジュール内でアセンブリの名前空間を指定できるように、Oracle 名前空間ディレクティブを追加できます。これを行うには、C# の `using` 文または Visual Basic の `Imports` 文をコード・ファイルの先頭または先頭付近に追加します。

注意： 参照を追加すると、アプリケーション内で名前空間を使用できるようになります。名前空間ディレクティブをアプリケーション・コードに追加することで、名前空間が明確になり、スコープの追加が可能になります。

Oracle 名前空間ディレクティブを追加するには、次の手順を実行します。

1. Form1 をアクティブにして、「**View**」メニューから「**Code**」を選択します。

   または、**[F7]** キーボード・ショートカットを使用することもできます。

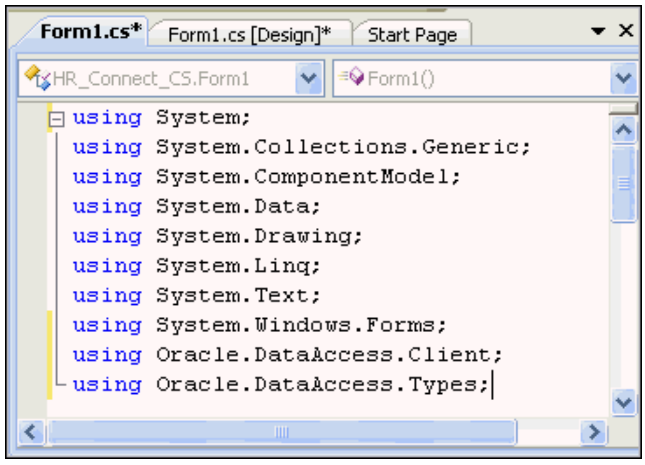


2. 使用している言語に応じて、次の文を宣言のリストに追加します。

   - **Visual C#**:

     他の `using` 文とともに、名前空間の前に追加します。

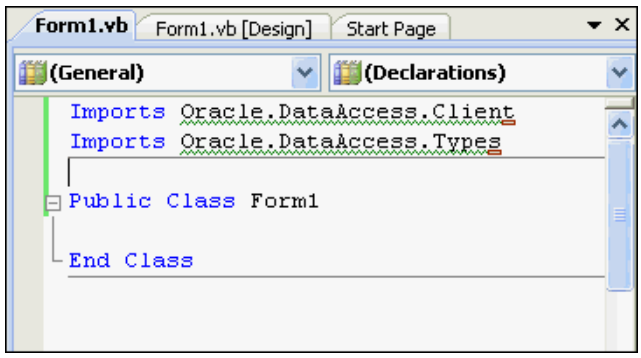
```
using Oracle.DataAccess.Client;
using Oracle.DataAccess.Types;
```



- **Visual Basic**:

  ファイルの先頭の宣言部に追加します。

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



3. 「**File**」メニューから「**Save**」を選択するか、**[Ctrl] + [S]** キーボード・ショートカットを使用して、変更を保存します。

ユーザー・インタフェースの設計

ユーザー・インタフェースは、デザイン・フォームにツールボックス・コントロールを追加して作成できます。このインタフェースは、ユーザーからの接続情報を受け入れます。

ツールボックス・コントロールを追加するには、次の手順を実行します。

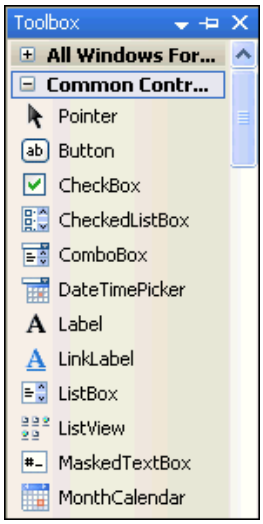
1. 「**View**」メニューから「**Designer**」を選択します。

   Form1 をまだ開いていない場合は、この操作で Form1 が設計ビューに表示されます。

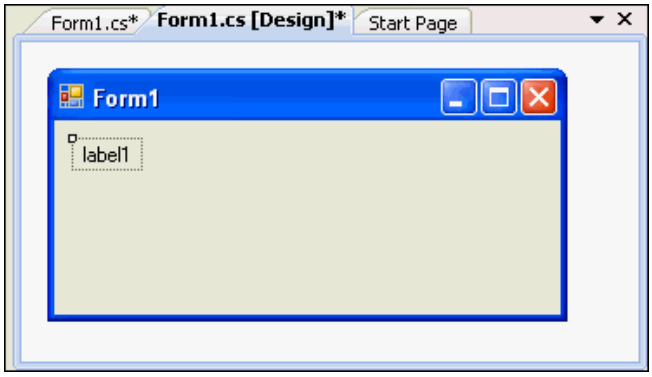
   後で、「Code」と「Designer」を何度も切り替えます。キーボード・ショートカットはそれぞれ **[F7]** と、**[Shift] + [F7]** です。

2. 「**View**」メニューから「**Toolbox**」を選択します。

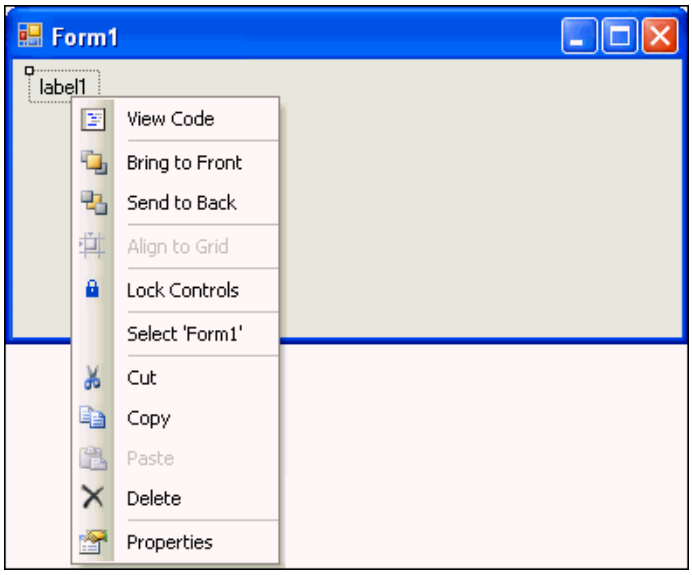
3. 「Toolbox」で「**Common Controls**」を開きます。



4. 「Toolbox」で**「Label」**を選択し、Form1 にドラッグします。



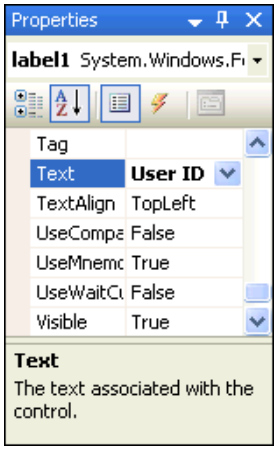
5. Form1 で**「label1」**を右クリックします。



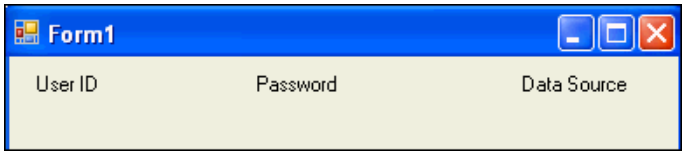
6. 「Properties」ウィンドウが表示されていない場合は、メニューから**「Properties」**を選択します。

   「Properties」ウィンドウが表示されます。

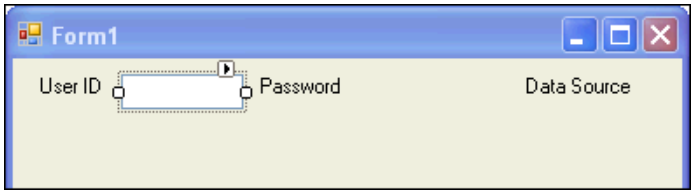
7. 「Properties」ウィンドウで Text プロパティを **label1** から **User ID** に変更します。



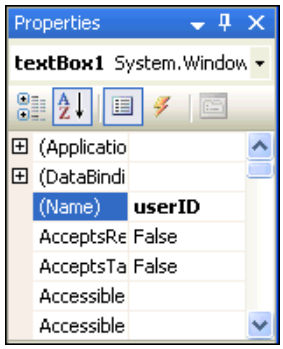
8. 手順 4 ～ 7 を 2 回繰り返し、さらに 2 つのラベルを Form1 に置きます。それらの Text プロパティを **Password** と **Data Source** に変更します。



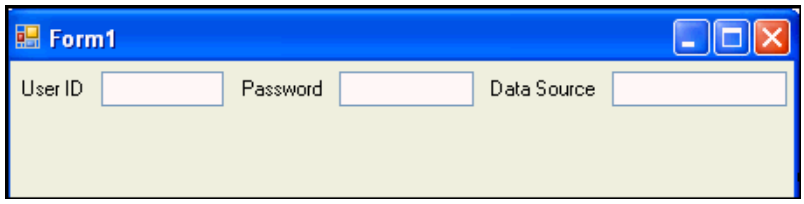
9. 「Toolbox」で「**TextBox**」を選択し、Form1 の「User ID」ラベルの隣にドラッグします。



10. 「Properties」ウィンドウで、Name プロパティを **userID** に変更します。

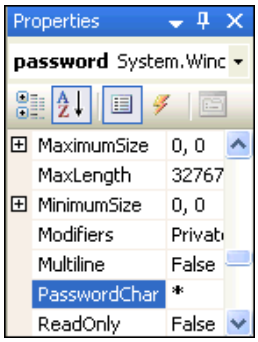


11. 手順 9 ～ 10 を 2 回繰り返し、さらに 2 つテキスト・ボックスを Form1 の既存のラベルの隣に配置します。それらの Name プロパティを **password** と **dataSource** に変更します。



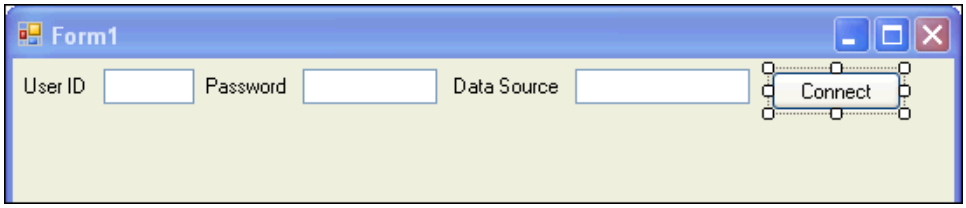
12. 「Password」ラベルの隣のテキストボックスを選択します。「Properties」ウィンドウで、PasswordChar プロパティまでスクロールして、このプロパティをアスタリスク（*）に設定します。

    これにより、入力中のパスワードがマスクされます。



13. 「Toolbox」で**「Button」**を選択し、Form1 にドラッグします。

    「Properties」ウィンドウで、このボタンの Text プロパティを**「button1」**から**「Connect」**に変更し、Name プロパティを **connect** に変更します。



14. 保存します。

15. ツールボックスを閉じます。

接続コードの記述

ここでは、ユーザー・インタフェースに入力された情報を使用してデータベースに接続するコードを記述します。

データベースに接続するには、接続オブジェクトを作成する必要があります。

データベースに接続するコードを記述するには、次の手順を実行します。

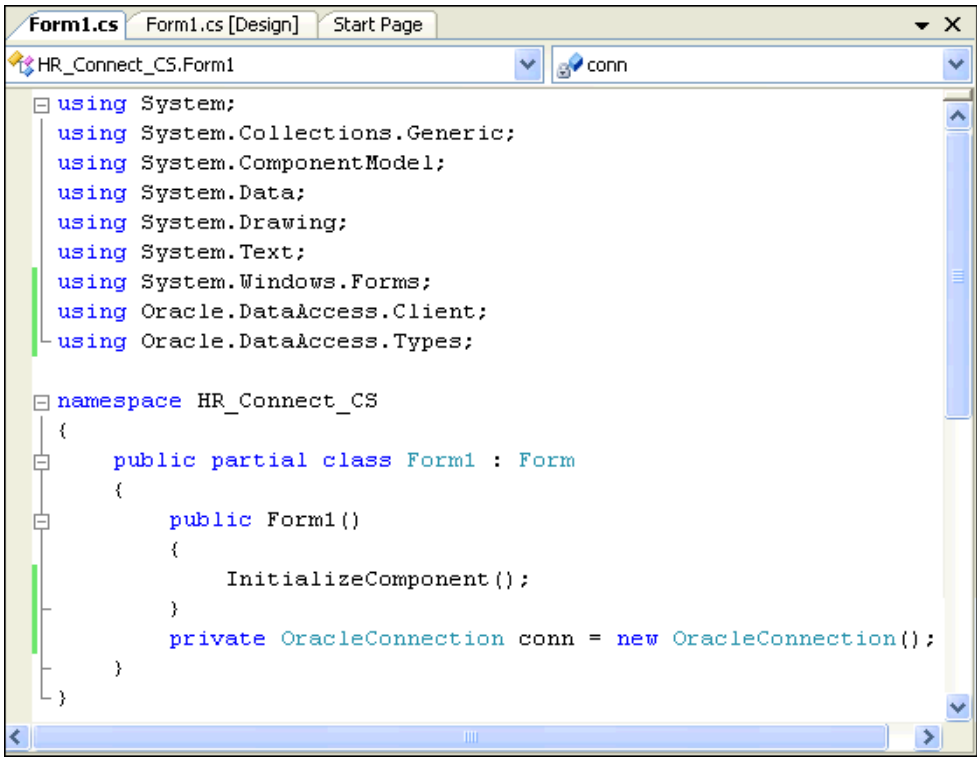
この手順を実行すると、ユーザーが Form1 のコントロールに入力したデータに基づいて、アプリケーションからデータベースに接続できるようになります。3-13 ページの「アプリケーションのコンパイルと実行」を参照してください。

1. **「View」**メニューから**「Code」**を選択します。

2. データベースの接続文字列をインスタンス化するために、次のコードを追加します。

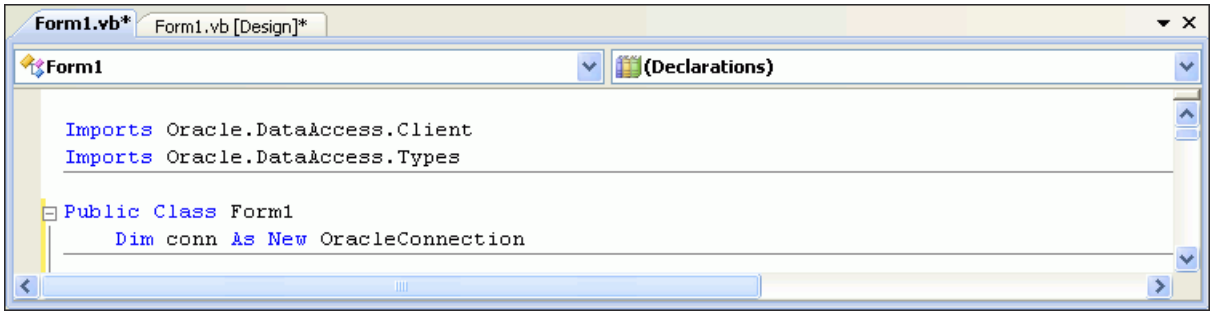
   Visual C#: 次のコードを使用して、Form1 クラスの public Form1() ブロックの直後にクラス変数 conn を追加します。

```
private OracleConnection conn = new OracleConnection();
```



   Visual Basic: 次のコードを使用して、Form1 クラス宣言に conn クラス変数を追加します。

```
Public Class Form1
   Dim conn As New OracleConnection
```



3. 変更を保存します。
4. 「**View**」メニューをクリックして「**Designer**」を選択し、「Designer」ビューに変更します。

5. Form1 の「**Connect**」ボタンをダブルクリックし、connect_Click() メソッドのコード・ウィンドウを開きます。

   次に示すコードを connect_Click() メソッドに挿入します。

   Visual C#:

```
conn.ConnectionString = "User Id=" + userID.Text +
   ";Password=" + password.Text +
   ";Data Source=" + dataSource.Text + ";";
conn.Open();
```

   Visual Basic:

```
conn.ConnectionString = "User Id=" + userID.Text & _
  ";Password=" + password.Text & _
  ";Data Source=" + dataSource.Text
conn.Open()
```

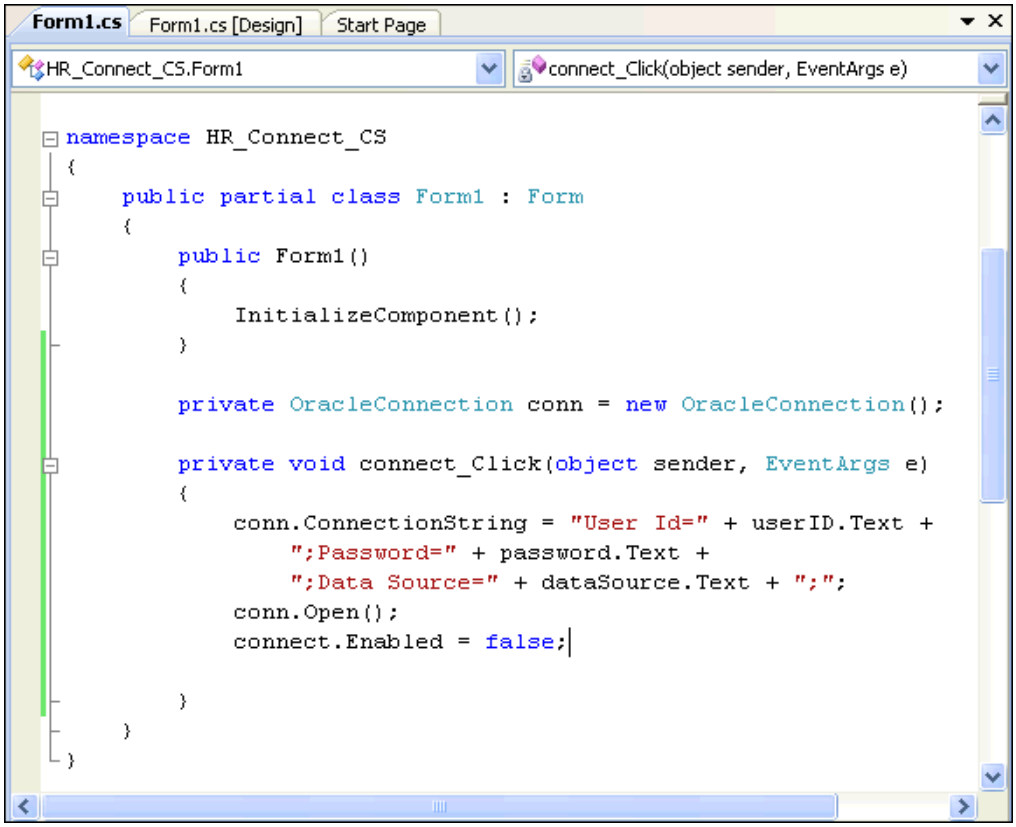
   注意：接続をオープンするには、ユーザーが入力した User ID、Password および Data Source の値から接続を作成する必要があります。Open() メソッドにより、実際の接続が作成されます。

6. connect_Click() メソッドの最後に次のコードを挿入し、このボタンの Enabled 属性を false に設定します。

   これにより、「Connect」ボタンは無効になります。正しく接続できた後は、このようにすることをお薦めします。

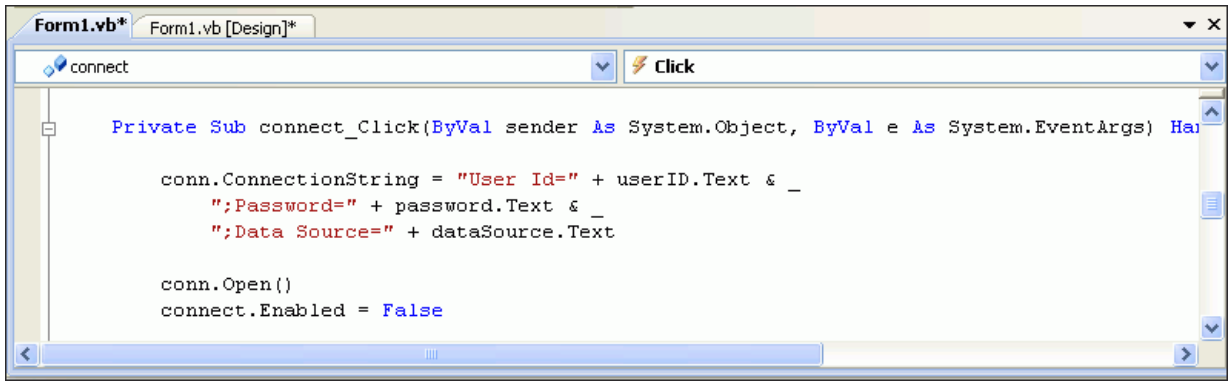
   Visual C#:

```
connect.Enabled = false;
```



Visual Basic:

```
connect.Enabled = false
```



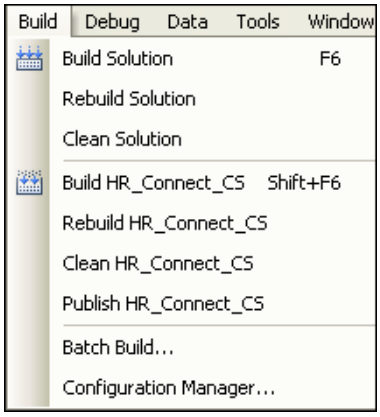
これで、Oracle Database に接続できるアプリケーションを記述できました。この後の項では、このアプリケーションの使用方法を説明します。

アプリケーションのコンパイルと実行

この項では、前の項で作成したアプリケーションをコンパイルして実行する方法を説明します。

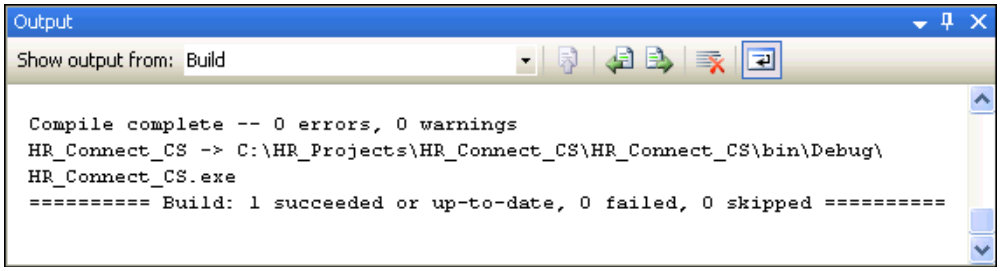
アプリケーションをコンパイルして実行するには、次の手順を実行します。

1. 「**Build**」メニューから「**Build Solution**」を選択します。



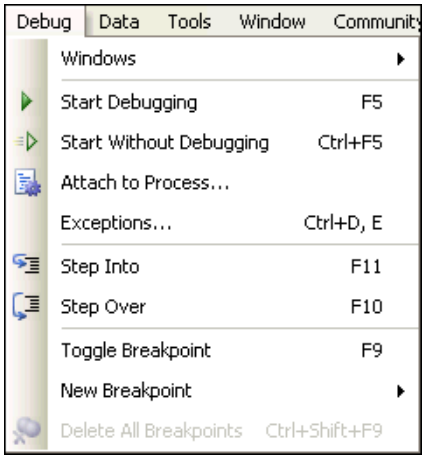
2. 「Output」ウィンドウにエラーが表示されておらず、「**View**」メニューから使用できることを確認します。

   標準的な出力結果は次のとおりです。



3. エラーがある場合は、「**View**」メニューから「**Error List**」を選択し、エラーを修正します。

4. 「**Debug**」メニューから「**Start Without Debugging**」を選択して、アプリケーションを実行します。

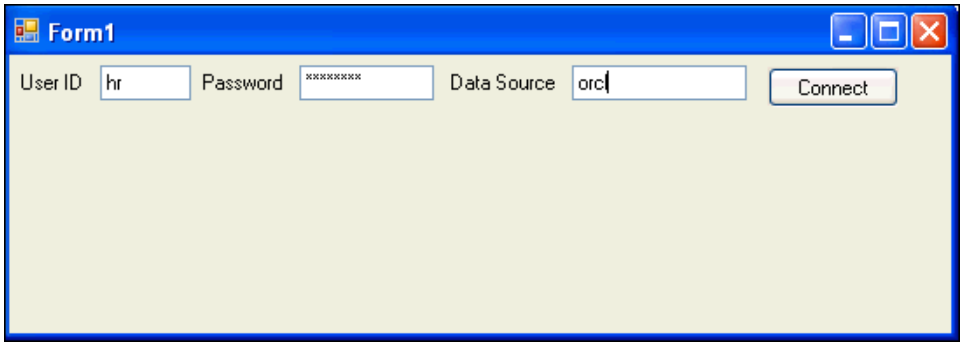


5. Form1 アプリケーションの「User ID」、「Password」および「Data Source」を入力します。

   「**Connect**」をクリックします。

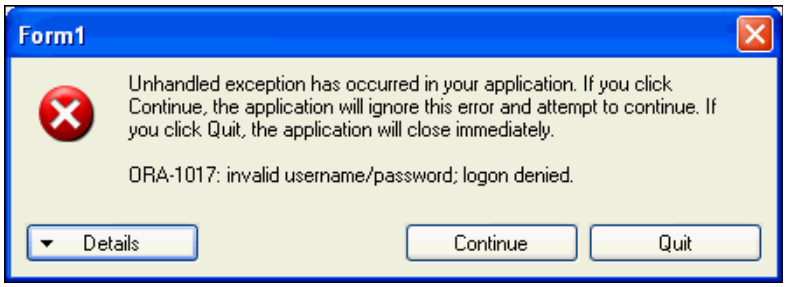
   これにより、アプリケーションで `tnsnames.ora` ファイルが使用されます。2-10 ページの「NET 接続の別名の構成」を参照してください。

   接続がオープンされたら、「Connect」ボタンは無効になります。これで、Oracle Database インスタンスへの接続が正しく実装されました。



エラー処理

アプリケーションでは、実行時エラーを正常に処理できる必要があります。たとえば、間違ったパスワードを使用してログインしようとすると、ここまでに開発したアプリケーションではデータベース接続を確立できず、「`ORA-1017: ユーザー名 / パスワードが無効です。ログオンは拒否されました。`」という未処理例外エラーで終了します。



「**Start Without Debugging**」を再度選択して、別のパスワードでログインを試行する必要があります。

エラー処理では、プログラム実行の標準フローとは異なる状況が発生した場合に、その対処方法を決定します。Oracle Data Provider for .NET には、エラーを処理およびサポートする次の 3 つのクラスが含まれています。

- `OracleError` クラスは、Oracle により報告される警告またはエラーを表します。
- `OracleErrorCollection` クラスは、Oracle Data Provider for .NET によりスローされるすべてのエラーの集合を表します。これは、`OracleError` のリストを保持する単純な `ArrayList` です。
- `OracleException` クラスは、Oracle Data Provider for .NET でエラーが発生したときにスローされる例外を表します。各 `OracleException` オブジェクトの「`Error`」プロパティには、エラーまたは警告を示す 1 つ以上の `OracleError` オブジェクトが含まれます。

Try-Catch-Finally ブロック構造の使用

.NET 言語では、エラー処理に Try-Catch-Finally ブロック構造を使用します。この構造では、Try コードがメイン・コードとなり、アプリケーションで実現する目標になります。次の 2 つの項で説明するように、Catch コードで様々な種類のエラーが取得されます。最後にある Finally ブロックは常に実行されます。

Finally ブロックには、接続をクローズして破棄する `Dispose` メソッドが含まれることがよくあります。Finally ブロックに `Dispose` メソッドを記述すると、Try-Catch-Finally ブロックが完了した後、データベース接続は常にクローズされます。アプリケーションでのデータベース・アクセスが不要になった後にデータベース接続をクローズすることは、特にデータ・セキュリティなど、多くの理由で重要です。

クローズされているデータベース接続をクローズしようとしてもエラーは発生しません。この試行は適切でないように思われます。しかし、`Dispose()` を Finally コード・ブロックに置くことで、接続が確実にクローズされるようになります。

次の項では、一般的なエラーでの Try-Catch-Finally ブロック構造の使用方法を説明します。その後の項では、Oracle エラーでの使用方法を説明します。

一般的なエラーの処理

この項では、Try-Catch-Finally ブロックを使用して一般的なエラーを処理する方法を説明します。

一般的なエラーを処理するには、次の手順を実行します。

1. Try-Catch-Finally 構文の実装を追加して、`Form1` の `connect_Click()` メソッドのコードを変更します。

   新しいコードは太字で示しています。

   Visual C#:

```
private void connect_Click(object sender, EventArgs e)
{
  conn.ConnectionString = "Data Source=ORCL;User Id="
     + userID.Text + ";Password=" + password.Text + ";";
  try
  {
    conn.Open();
    connect.Enabled = false;
  }
  catch (Exception ex)
  {
    MessageBox.Show(ex.Message.ToString());
  }
```

```
  finally
  {
    conn.Dispose();
  }
}
```

または、次のように using キーワードを使用して、有効でなくなった接続を破棄する C# 構文を使用することもできます。

```
using (OracleConnection conn = new OracleConnection())
{
  conn.Open();
  // application code
  ...
}
```

Visual Basic:

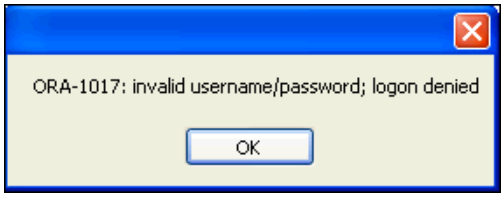
```
Try
  conn.Open()
  connect.Enabled = false

Catch ex As Exception
  MessageBox.Show(ex.Message.ToString())

Finally
  conn.Dispose()
End Try
```

2. 「**Build**」メニューから「**Rebuild Solution**」を選択します。

   エラーがないことを確認します。

3. 「**Debug**」メニューから「**Start Without Debugging**」を選択します。

4. 3-13 ページの「アプリケーションのコンパイルと実行」の手順に従ってアプリケーションを再度実行し、間違ったパスワードを使用して接続を試行します。今度はアプリケーションでエラーが取得され、ポップアップ・ウィンドウに「ORA-1017: ユーザー名 / パスワードが無効です。ログオンは拒否されました。」と表示されます。



一般的な Oracle エラーの処理

次に示す完成した Try-Catch-Finally ブロック・コードでは、OracleException がない場合は、最初の Catch 文のブランチがスキップされます。2 番目の Catch 文のブランチで、その他のすべての Exception が取得されます。

最初の Catch 文には Case 文が含まれており、これにより一般的なデータベース・エラーがトラップされ、ユーザーにわかりやすい方法で表示されます。

2 番目の Case 文では、データベースにアクセスできない場合に、特定の OracleException が取得されます。

特定のエラーを処理するには、次の手順を実行します。

1. データベース・インスタンスを停止します。付録 A「Oracle Database インスタンスの起動および停止」を参照してください。

2. 次の太字で示されている Catch OracleException ブロックを、connect_Click() メソッドに以前に追加した Catch Exception ブロックの前に追加します。

Visual C#:

```
try
  {
    conn.Open();
    connect.Enabled = false;
  }
catch (OracleException ex)
{
  switch (ex.Number)
  {
    case 1:
      MessageBox.Show("Error attempting to insert duplicate data.");
      break;
    case 12560:
      MessageBox.Show("The database is unavailable.");
      break;
    default:
      MessageBox.Show("Database error: " + ex.Message.ToString());
       break;
  }
}
 catch (Exception ex)
   {
     MessageBox.Show(ex.Message.ToString());
   }
finally
  {
    conn.Dispose();
  }
}
```

Visual Basic:

```
Try
  conn.Open()
  connect.Enabled = false

Catch ex As OracleException ' catches only Oracle errors
  Select Case ex.Number
    Case 1
      MessageBox.Show("Error attempting to insert duplicate data.")
    Case 12560
      MessageBox.Show("The database is unavailable.")
    Case Else
      MessageBox.Show("Database error: " + ex.Message.ToString())
  End Select


Catch ex As Exception
  MessageBox.Show(ex.Message.ToString())

Finally
  conn.Dispose()
End Try
```

3. 3-13 ページの「アプリケーションのコンパイルと実行」の手順に従ってアプリケーションを再度コンパイルして実行します。

   ORA-12560 エラーが、エラー番号なしで「The database is unavailable」としてポップアップ・ウィンドウに表示されることを確認します。

4. データベース・インスタンスを再起動します。付録 A「Oracle Database インスタンスの起動および停止」を参照してください。

4

Oracle Data Provider for .NET での取得と更新

この章の内容は次のとおりです。

コマンド・オブジェクトの使用

データベース内のデータを表示、編集、挿入または削除するには、SQL コマンド、ストアド・プロシージャまたは表名を指定して、OracleCommand オブジェクトにリクエストをカプセル化する必要があります。OracleCommand オブジェクトは、リクエストを作成してデータベースに送信し、結果を戻します。

コマンド・オブジェクトを使用するには、次の手順を実行します。

1. 第 3 章「ODP.NET による単純な .NET アプリケーションの作成」で作成したフォーム・アプリケーション HR_Connect_*xx* から、Form1.*xx* のコピーを 2 つ作成します。コピーを作成する方法については、付録 B「フォームのコピー」を参照してください。

   コピーに Form2.cs または Form2.vb、および Form3.cs または Form3.vb と名前を付けます。最初のコピーは、この章の前半で使用します。2 つ目のコピーは、この章の後半で使用します。

2. Form2.cs または Form2.vb を開きます。

   コード・ファイルの名前は変更しましたが、プロジェクト内の実際のフォーム・コントロールの名前は変更していないため、デザイナのフォームはまだ Form1 となっていることに注意してください。

3. SQL 問合せを表す文字列を作成し、try 文の本体に追加します。

   新しいコードは太字で示しています。

   Visual C#:

```
try
{
    conn.Open();
    connect.Enabled = false;

    // SQL Statement
    string sql = "select department_name from departments"
       + " where department_id = 10";
}
```

   Visual Basic:

```
Try
      conn.Open()
      connect.Enabled = False

     Dim sql As String = "select department_name from departments" & _
       "where department_id = 10"
```

4. 新しい sql 変数を使用して OracleCommand オブジェクトを作成し、テキスト・コマンドが実行されるように CommandType プロパティを設定します。

   Visual C#:

```
try
{
    conn.Open();
    connect.Enabled = false;

    // SQL Statement
    string sql = "select department_name from departments"
       + " where department_id = 10";

OracleCommand cmd = new OracleCommand(sql, conn);
cmd.CommandType = CommandType.Text;
}
```

Visual Basic:

```
Try
      conn.Open()
      connect.Enabled = False

      Dim sql As String = "select department_name from departments" & _
         "where department_id = 10"

Dim cmd As New OracleCommand(sql, conn)
cmd.CommandType = CommandType.Text
```

5. 実行した内容を保存します。

データの取得 : 単純な問合せ

この項では、データベースからデータを取得する方法を説明します。

OracleCommand オブジェクトの ExecuteReader() メソッドにより OracleDataReader オブジェクトが戻されます。これにアクセスすることでフォームに結果を表示できます。このアプリケーションでは、ListBox を使用して結果を表示します。

データを取得するには、次の手順を実行します。

1. 次に示すコードを connect_Click() メソッドの Try ブロックの最後に追加して、OracleDataReader オブジェクトを作成します。

   これにより、問合せ結果を読み取ることができます。

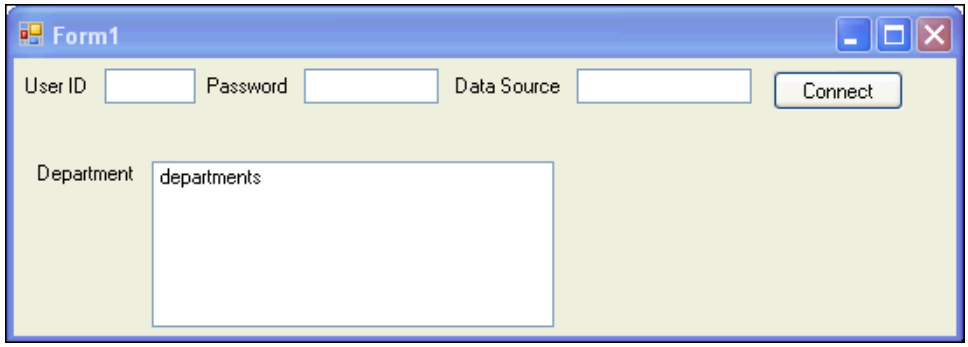
   Visual C#:

   ```
   OracleDataReader dr = cmd.ExecuteReader();
   dr.Read();
   ```

   Visual Basic:

   ```
   Dim dr As OracleDataReader = cmd.ExecuteReader()
   dr.Read()
   ```

2. 設計ビューで Form1 を開きます。「**View**」メニューから「**Designer**」を選択します。
3. 「**View**」メニューから「**Toolbox**」を選択します。
4. 「Toolbox」から「**Label**」を選択し、Form1 にドラッグします。
5. 「**View**」メニューから「**Properties Window**」を選択します。
6. 「Properties」ウィンドウで、ラベルの「**Text**」を Department に変更します。
7. 「Toolbox」の「Window」フォームから、「**ListBox**」を選択し、Form1 にドラッグします。
8. 「Properties」ウィンドウの「Design」で、「**Name**」を departments に変更します。



9. 問合せ結果からデータを取得するためのアクセッサ型メソッドを追加します。

   「Connect」ボタンをダブルクリックして connect_click() メソッドを編集し、次に示すコードを Try ブロックの最後に追加します。

   Visual C#:

   ```
   departments.Items.Add(dr.GetString(0));
   ```

   Visual Basic:

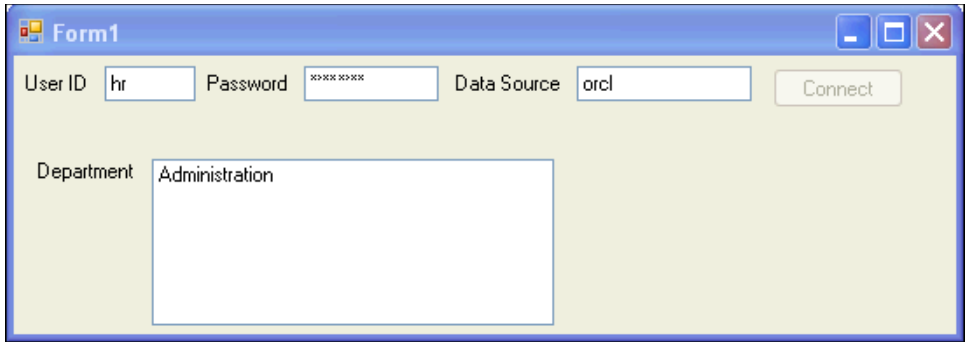
   ```
   departments.Items.Add(dr.GetString(0))
   ```

   GetString などの型指定されたアクセッサは、ネイティブの .NET データ型およびネイティブの Oracle データ型を戻します。結果セットのどの列を戻すかは、アクセッサに渡されたゼロベースの序数で指定します。

10. アプリケーションをビルドして保存します。

11. アプリケーションを実行します。ログインおよびデータソースを入力します。

    接続すると、「Department」リスト・ボックスに Administration と表示されます。これは HR スキーマに含まれる部門番号 10 の正しい名前で、SELECT 文でリクエストしたものです。



データの取得 : バインド変数

バインド変数は SQL 文内のプレースホルダです。データベースでは SQL 文を受信すると、その文がすでに実行されたことがありメモリーに格納されているかどうかが確認されます。その文がメモリーに存在する場合、Oracle Database ではその文を再利用でき、文の解析と最適化のタスクがスキップされます。バインド変数を使用すると、異なる入力値でも文の再利用が可能です。また、バインド変数を使用すると、データベースの問合せパフォーマンスが向上するだけでなく、入力に含まれるリテラル引用符の特別な処理が不要になり、SQL インジェクション攻撃から保護することができます。

次のコードは、バインド変数を使用せずに、文の WHERE 句に値 10 を指定する標準的な SELECT 文です。

```
SELECT department_name FROM departments WHERE department_id = 10
```

次のコードは、数値をバインド変数 :department_id で置き換えたものです。バインド変数識別子は、常に 1 つのコロン（:）で始まります。

```
SELECT department_name FROM departments WHERE department_id = :department_id
```

バインド変数は UPDATE、INSERT および DELETE 文でも使用でき、ストアド・プロシージャでも使用できます。次のコードは、UPDATE 文でバインド変数を使用する方法を示しています。

```
UPDATE departments SET department_name = :department_name
  WHERE departname_id = : department_id
```

詳細は、4-12 ページの「データの挿入、削除および更新」を参照してください。

.NET コードで各バインド変数を表すには、OracleParameter クラスを使用できます。OracleParameterCollection クラスには、各文の OracleCommand オブジェクトと関連付けられた OracleParameter オブジェクトが含まれています。OracleCommand クラスは、SQL 文をデータベースに渡し、結果をアプリケーションに戻します。

変数は、OracleCommand プロパティ BindByName の設定（デフォルトは false）によって、位置または名前でバインドできます。

- 位置指定によるバインド

  Add() メソッドを使用して、SQL 文またはストアド・プロシージャで記述されるのと同じ順序でパラメータを OracleParameterCollection に追加する必要があります。

- 名前指定によるバインド

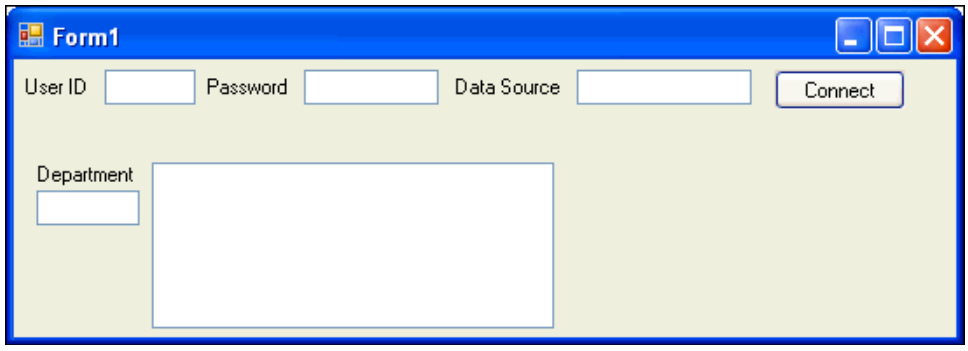
  任意の順序でパラメータをコレクションに追加できます。ただし、パラメータ・オブジェクトの ParameterName プロパティを、ストアド・プロシージャで宣言されているバインド変数識別子と同じ名前に設定する必要があります。

バインド・モード（位置指定または名前指定）の他に、.NET 開発者は Direction、OracleDbType、Size および Value の各プロパティを、各パラメータ・オブジェクトに設定します。

- **Direction** バインド変数は、出力、入力または入出力パラメータとして使用できます。Direction プロパティは、各パラメータの向きを示します。Direction プロパティのデフォルト値は Input です。
- **OracleDbType** プロパティは、パラメータが number、date、VARCHAR2 などのいずれであるかを示します。
- **Size** は、VARCHAR2 などの可変長データ型のパラメータで保持できる最大データ・サイズを示します。
- **Value** には、文の実行前（入力パラメータ）、実行後（出力パラメータ）または実行前後の両方（入出力パラメータ）のパラメータ値が保持されます。

バインド変数を使用してデータを取得するには、次の手順を実行します。

1. Departments という名前の ListBox を右側に移動します。
2. 「**View**」メニューから「**Toolbox**」を選択します。
3. 「**Toolbox**」から「**TextBox**」を選択し、Form1 の Department という名前のラベルの下にドラッグします。
4. 「**View**」メニューから「**Properties Window**」を選択します。
5. 「Properties」ウィンドウで、「**Name**」を departmentID に変更します。



6. 次に示すコードを connect_Click() メソッドの Try ブロックに追加して、バインド変数が使用されるように SELECT 文を変更します。

   変更されたコードまたは新しいコードは太字で示しています。

   Visual C#:

```
string sql = "select department_name from departments where department_id = " +
  ":department_id";
OracleCommand cmd = new OracleCommand(sql, conn);
cmd.CommandType = CommandType.Text;
OracleParameter p_department_id = new OracleParameter();
p_department_id.OracleDbType = OracleDbType.Decimal;
p_department_id.Value = departmentID.Text;
cmd.Parameters.Add(p_department_id);

OracleDataReader dr = cmd.ExecuteReader();
dr.Read();

departments.Items.Add(dr.GetString(0));
```

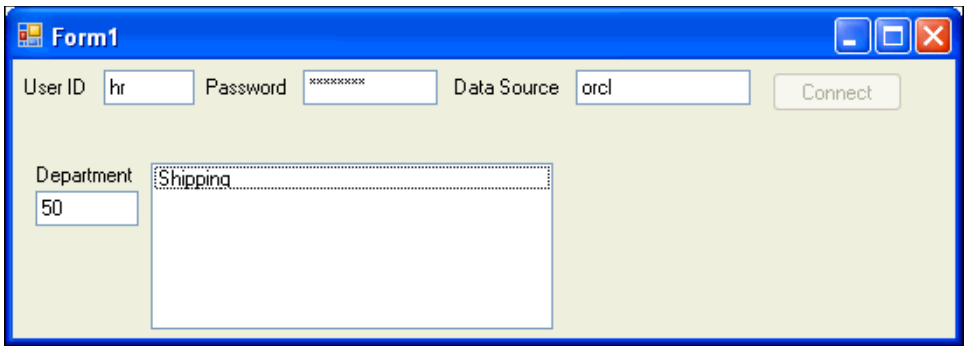
   Visual Basic:

```
Dim sql As String = "select department_name from departments where" & _
  "department_id= ":department_id"
Dim cmd As OracleCommand = New OracleCommand(sql, conn)
cmd.CommandType = CommandType.Text
Dim p_department_id as OracleParameter = new OracleParameter()
p_department_id.OracleDbType = OracleDbType.Decimal
p_department_id.Value = departmentID.Text
cmd.Parameters.Add(p_department_id)

Dim dr As OracleDataReader = cmd.ExecuteReader()
dr.Read()

departments.Items.Add(dr.GetString(0))
```

   このコードでは、パラメータ・オブジェクトは OracleDbType プロパティを設定しますが、Direction プロパティにはデフォルト値 Input を使用するため、設定は不要です。オブジェクトは入力パラメータであり、データ・プロバイダは値からサイズを判断できるため、Size プロパティを設定する必要はありません。

7. アプリケーションを保存して実行します。
8. ログイン情報と、HR スキーマの代表的な部門番号（50 など）を入力します。
9. 「Connect」をクリックします。

   部門 ID に対応する部門名がアプリケーションから戻されます。



データの取得 : 複数の値

データベースから複数の値を取得することが必要になる場合はよくあります。複数の列および複数の行の値を取得するには、DataReader オブジェクトを使用できます。次の例で、複数の列や複数の行に対する問合せについて考えてみます。

```
SELECT department_id, department_name, manager_id, location_id
  FROM departments
  WHERE department_id < 100
```

DataReader オブジェクトから複数の行を処理するには、ループ構造が必要です。また、複数の行を表示できるコントロールが役立ちます。OracleDataReader オブジェクトは前進専用で読取り専用のカーソルであるため、Windows Forms の DataGrid コントロールなどの、更新可能なコントロールまたは後方にスクロールできるコントロールにはバインドできません。ただし、OracleDataReader オブジェクトは ListBox コントロールと互換性があります。

複数の値を取得するには、次の手順を実行します。

1. connect_Click() メソッドの Try ブロックで、複数の行の結果セットが戻され、部門名を表示する read メソッドを囲む while ループを追加するように SQL 問合せを変更します。

   Visual C#:

   ```
   try
   {
     ...
   string sql = "select department_name from departments where department_id" +
     "< :department_id";
   ...
     while (dr.Read())
     {
       departments.Items.Add(dr.GetString(0));
     }
   }
   ```

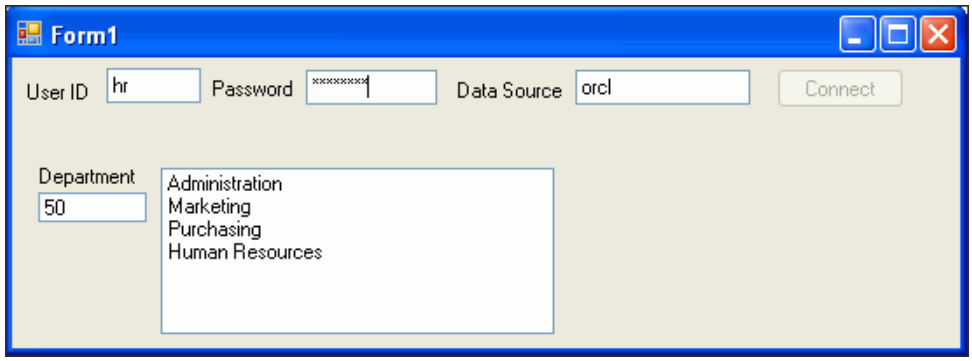
   Visual Basic:

   ```
   Try
     ...
     Dim sql As String = "select department_name from departments " & _
           "where department_id < :department_id"
   ...
     While (dr.Read())
       departments.Items.Add(dr.GetString(0))
     End While
   ```

2. アプリケーションを保存して実行します。
3. ログイン情報を入力し、部門に 50 と入力します。

4. 「**Connect**」をクリックします。

   問合せに対応する部門名が、アプリケーションから戻されます。

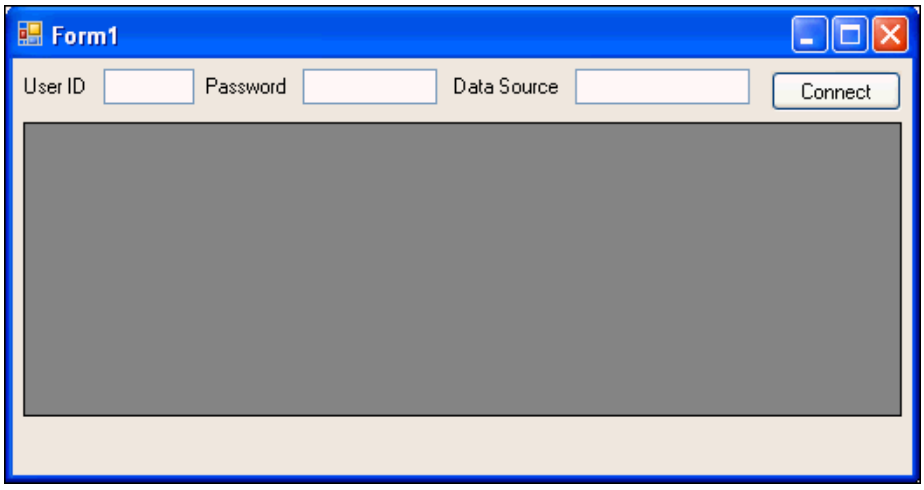


Oracle Data Provider for .NET での DataSet クラスの使用

`DataSet` クラスでは、メモリーに常駐するデータベース・データのコピーが提供されます。これは、リレーショナル・データまたは XML データを格納する 1 つ以上の表で構成されます。`OracleDataReader` オブジェクトとは異なり、`DataSet` は更新可能で後方にスクロールできます。

Dataset クラスを使用するには、次の手順を実行します。

1. 第 3 章で作成した Form1 のコピーを作成していない場合は Form1 をコピーし、付録 B「フォームのコピー」で説明するとおり、`Form3.vb` または `.cs` という名前を付けます。`Form1.xx` が Solution Explorer に表示されない場合は、「**Project**」メニューから「**Show All Files**」を選択します。
2. 「**View**」メニューから「**Designer**」ビューを選択します。
3. 「**View**」メニューから「**Toolbox**」を選択します。
4. 「**Toolbox**」から「**DataGridView**」を選択し、Form1 にドラッグします。
5. 「**View**」メニューから「**Properties Window**」を選択します。
6. 「**Properties**」ウィンドウで、データ・グリッド・ビューの「**Name**」を `departments` に変更します。



7. 「**View**」メニューから「**Code**」を選択します。

8. 次に示すように、コードの conn 宣言の直後に、クラス変数への変数宣言を追加します。

Visual C#:

```
public partial class Form1 : Form
{
  public Form1()
  {
      InitializeComponent();
  }
  private OracleConnection conn = new OracleConnection();
  private OracleCommand cmd;
  private OracleDataAdapter da;
  private OracleCommandBuilder cb;
  private DataSet ds;
...
```

Visual Basic:

```
Public Class Form1
    Dim conn As New OracleConnection
    Private cmd As OracleCommand
    Private da As OracleDataAdapter
    Private cb As OracleCommandBuilder
    Private ds As DataSet
```

9. connect_Click() メソッドの Try ブロックに、次のコードを追加します。

- データベースを問い合せるコード
- コマンドの問合せ結果を DataSet に埋め込むコード
- DataSet をデータ・グリッド（departments）にバインドするコード

Visual C#:

```
conn.Open();
connect.Enabled = false;

string sql = "select * from departments where department_id < 60";
cmd = new OracleCommand(sql, conn);
cmd.CommandType = CommandType.Text;

da = new OracleDataAdapter(cmd);
cb = new OracleCommandBuilder(da);
ds = new DataSet();

da.Fill(ds);

departments.DataSource = ds.Tables[0];
```

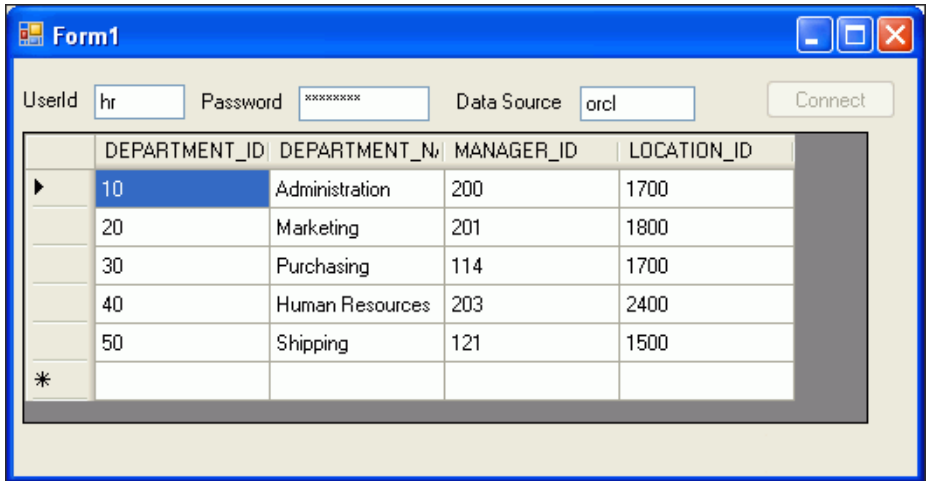
Visual Basic:

```
conn.Open()
connect.Enabled = False

Dim sql As String = "select * from departments where department_id < 60"
cmd = New OracleCommand(sql, conn)
cmd.CommandType = CommandType.Text

da = New OracleDataAdapter(cmd)
cb = New OracleCommandBuilder(da)
ds = New DataSet()
```

```
da.Fill(ds)

departments.DataSource = ds.Tables(0)
```

10. アプリケーションをビルドして保存します。
11. アプリケーションを実行し、ログインおよびデータソースを入力します。

    データベースに正しく接続されると、データ・グリッドに問合せ結果が移入されます。



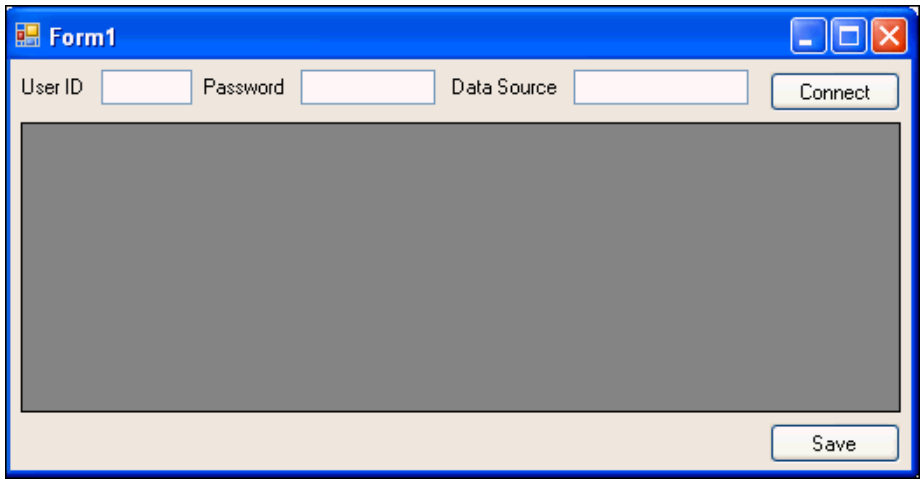
データベースの更新の有効化

この時点で、DataSet にはデータベース・データのクライアント・コピーが保持されています。この項では、クライアント・データの変更をデータベースに保存できるようにするボタンを追加します。その後の項では、データの更新、挿入および削除のテスト方法を説明します。

DataSet からデータベースにデータを保存できるようにするには、次の手順を実行します。

1. 「Toolbox」から「**Button**」を Form1 にドラッグ・アンド・ドロップします。
2. 「**Properties**」ウィンドウで、ボタンの「**Name**」を save に変更します。

   Text プロパティを Save に変更します。
3. 「**Properties**」ウィンドウの上部で、「**Events**」（稲妻のアイコン）をクリックします。イベントのリストでクリック・イベントを選択します。2 番目の列にイベント名 save_Click を入力します。



4. 「**View**」メニューから「**Code**」を選択します。
5. 次に示すように、データを更新するコードを save_Click() メソッドの本体に追加します。

   Visual C#:

   ```
   da.Update(ds.Tables[0]);
   ```

   Visual Basic:

   ```
   da.Update(ds.Tables(0))
   ```

   「Error List」にエラーが表示される場合があります。これらのエラーは、次の手順でコードを追加する表示されなくなります。
6. Form() メソッドまたは Form1_Load メソッドに、次のコードを追加します。

   Visual C#:

   ```
   public Form1()
   {
       InitializeComponent();
       save.Enabled = false;
   }
   ```

   Visual Basic:

   ```
   Private Sub Form1_Load(ByVal sender As System.Object, & _
       ByVal e As System.EventArgs) Handles MyBase.Load
       save.Enabled = false
   ```

7. 次のように、connect_Click() メソッドの Try ブロックに、「**Save**」ボタンを有効にするコードを追加します。

   Visual C#:

   ```
   conn.Open();
    ...
   departments.DataSource = ds.Tables[0];

   save.Enabled = true;
   ```

   Visual Basic:

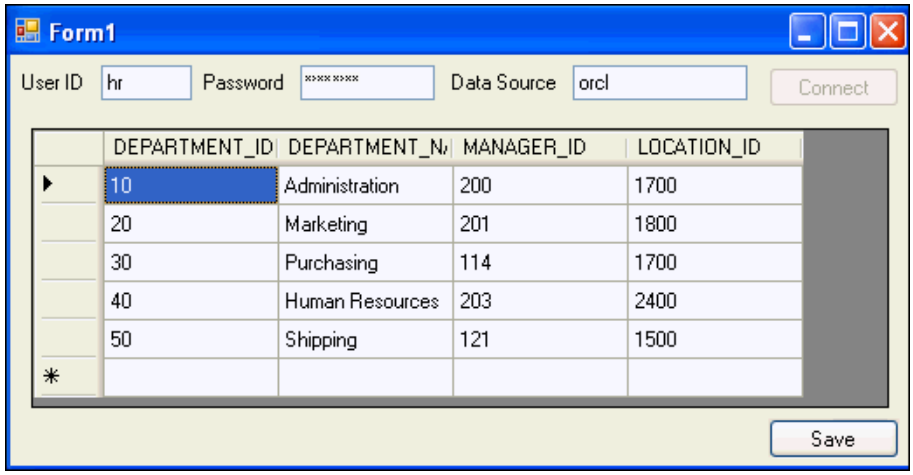
   ```
   conn.Open()
   ...
   departments.DataSource = ds.Tables(0)

   save.Enabled = True
   ```

8. conn.Dispose() コールを connect_Click() メソッドの Finally ブロックから削除します。

   注意 : この例で前に使用したコードでは、接続を破棄またはクローズするためにこのメソッドが必要でした。しかし、コードを変更したため、問合せ結果が戻された後も接続をオープン状態のままにし、エンド・ユーザーによるデータ変更がデータベースに伝播されるようにする必要があります。一般的なオーバーライド・コールである components.Dispose() は、すでに Form1 の定義に含まれています。
9. アプリケーションをビルドして保存します。

10. アプリケーションを実行し、ログインおよびデータソースを入力します。

    データベースに正しく接続されると、データ・グリッドに問合せ結果が移入されます。

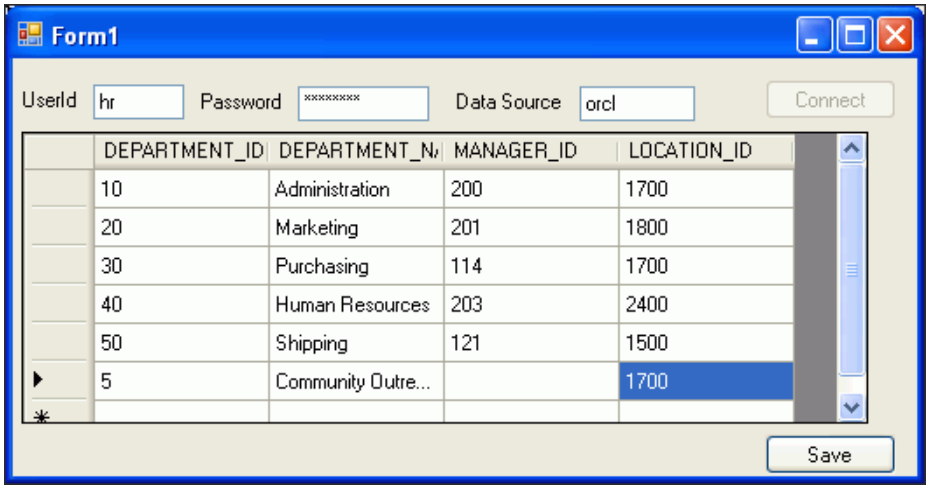


データの挿入、削除および更新

この項では、新しいアプリケーションを使用してデータベースのデータを直接操作する方法を説明します。

データを挿入、削除および更新するには、次の手順を実行します。

1. 前の項で作成したアプリケーションを実行し、ログインおよびデータソースを入力して、データベースに接続します。
2. データ・グリッドの一番下にある * プロンプトで、新しいレコードを入力します。
   - 「DEPARTMENT_ID」に 5 と入力します。
   - 「DEPARTMENT_NAME」に Community Outreach と入力します。
   - 「MANAGER_ID」には値を入力せず、そのままにします。
   - 「LOCATION_ID」に 1700 と入力します。

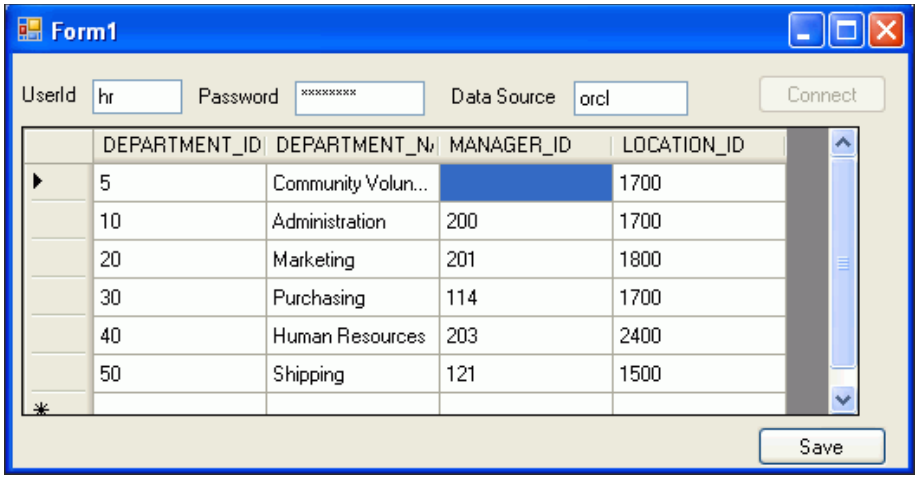


3. 「**Save**」をクリックします。
4. アプリケーションを閉じ、新しいレコードが保存されているかどうかを確認します。

5. アプリケーションを再度実行し、データベースに接続します。

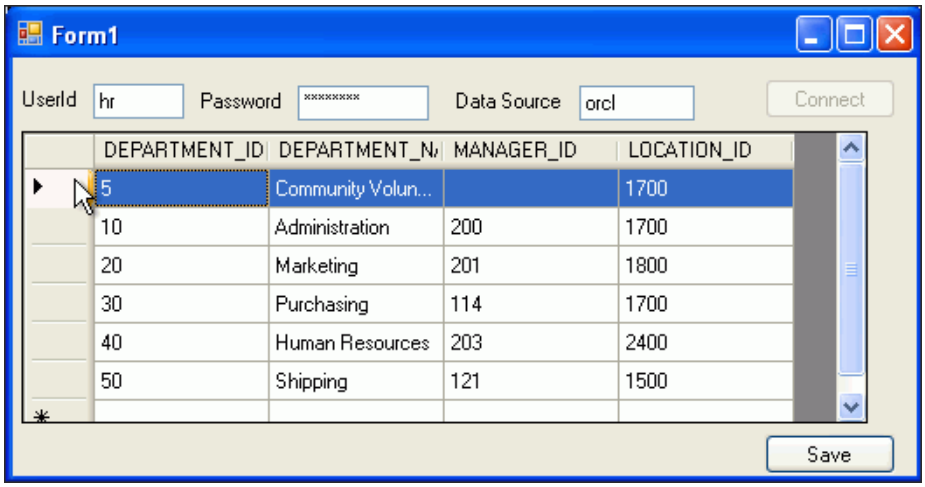
   DEPARTMENT_ID が番号順に表示され、新しい部門が DEPARTMENTS 表の先頭にあることを確認します。

6. 部門名を Community Volunteers に変更して、「**Save**」ボタンをクリックします。



7. 手順 4 を繰り返します。アプリケーションを再度実行し、データベースに接続して、部門名が変更されていることを確認します。

8. 変更したばかりのレコード全体を選択し（一番左の列にあるカーソル・アイコンをクリック）、**[Delete]** キーを使用してこのレコードを削除します。「**Save**」ボタンをクリックします。



9. 手順 4 を繰り返します。アプリケーションを再度実行し、データベースに接続して、新しいレコードが DEPARTMENTS 表に含まれていないことを確認します。

10. アプリケーションを閉じます。

5

Oracle Developer Tools for Visual Studio の使用

この章の内容は次のとおりです。

Oracle Developer Tools の使用

Oracle Developer Tools for Visual Studio（ODT）は、Visual Studio 用に緊密に統合されたアドインです。ODT により Server Explorer に組み込まれる拡張機能を使用することで、表、索引、制約、データ接続などのデータベース・スキーマ・オブジェクトを自動作成できます。また、アプリケーション・コードも自動生成できます。

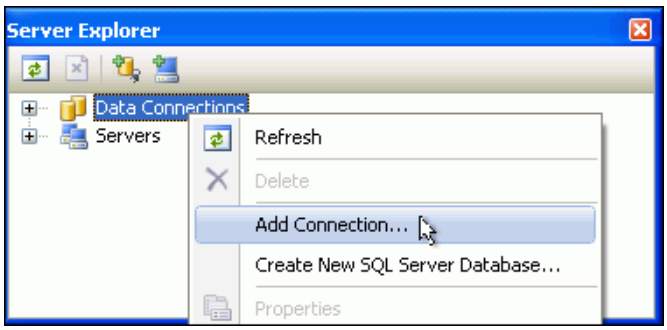
参照： 1-2 ページの「Oracle Developer Tools for Visual Studio の概要」

Oracle Database への接続

この項では、データベース・スキーマ・オブジェクトを自動的に作成または変更するために、Server Explorer を使用して Oracle Database に接続する方法を説明します。

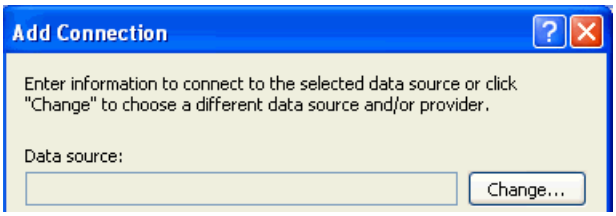
データベースに接続するには、次の手順を実行します。

1. 「View」メニューから**「Server Explorer」**を選択します。
2. Server Explorer で、**「Data Connections」**を右クリックします。
3. **「Add Connection」**を選択します。



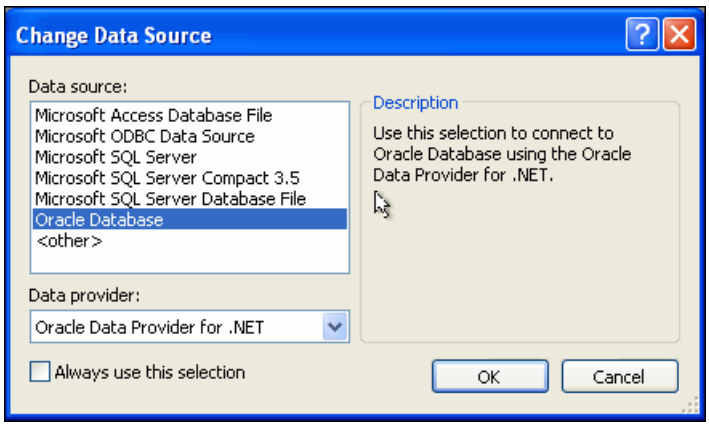
4. 「Add Connection」ウィンドウが表示されたら、「Data source」に「Oracle Database (Oracle ODP.NET)」と表示されているかどうかを確認します。

   表示されている場合は、手順 6 に進みます。



   「Data source」に「Oracle Database (Oracle ODP.NET)」と表示されていない場合は、**「Change」**を選択します。

   「Change Data Source」ウィンドウが表示されます。



5. 「Oracle Database」を選択してから、「Oracle Data Provider for .NET」を選択します。

6. 「Add Connection」ウィンドウの「Connection Details」タブで、次の情報を入力します。

   Data source name: リモート・データベース・インスタンスの別名 `orcl` などを使用します。

   同じコンピュータ上のデータベースに接続する場合は、`Local Database` を使用します。

   「Use a specific user name and password」オプションを選択します。

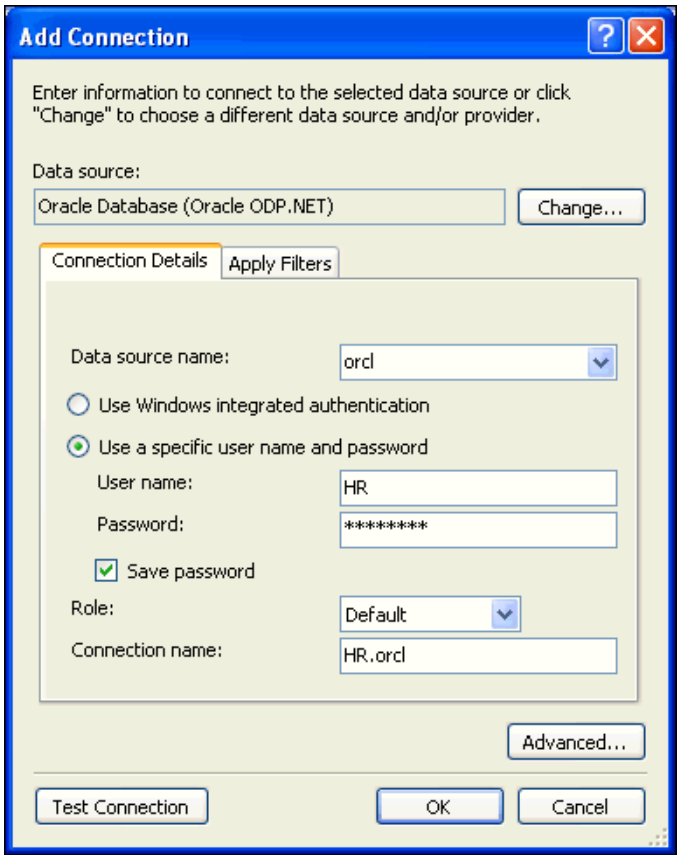
   「User name」に `HR` と入力します。

   「Password」に、`hr` アカウントのロック解除と設定を行ったときに作成したパスワードを入力します。

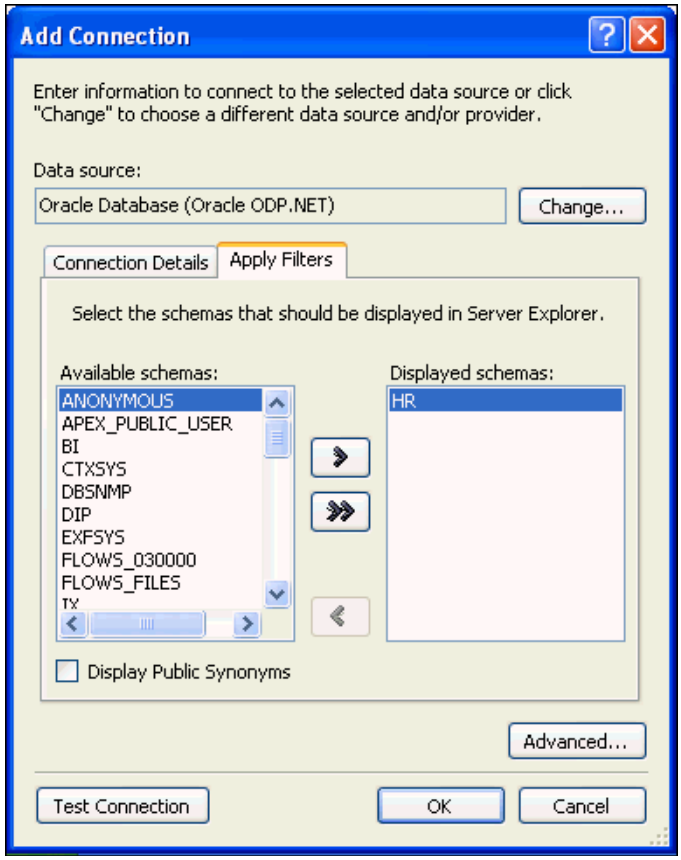
   後続のセッション用にパスワードを保存するには、**「Save password」**ボックスを選択します。

   「Role」が `Default` に設定されていることを確認します。これにより、ユーザー `hr` に付与されているデフォルトのロールが参照されます。

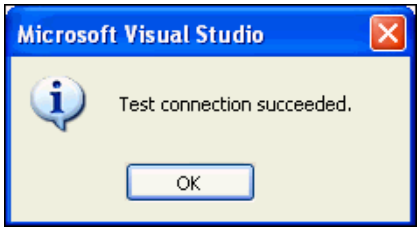
   「Connection name」は**「Data source name」**および**「User name」**の値から自動生成されます。この演習では、**HR.orcl** になります。



7. 「**Apply Filters**」タブをクリックし、HR スキーマが「**Displayed schemas**」列に含まれていることを確認します。データ接続のスキーマ・カテゴリ・ノードを開くと、「**Apply Filters**」タブで選択したスキーマ・オブジェクト（表、ビューなど）のみが表示されます。



8. 「**Test connection**」をクリックします。

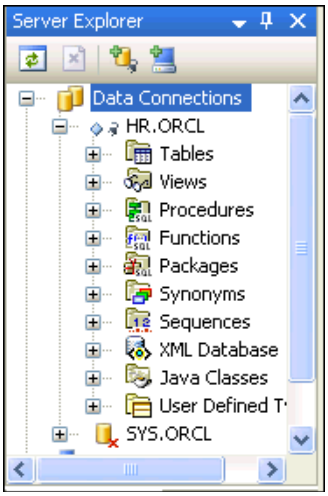


テストは成功するはずです。「**OK**」をクリックします。

テストが失敗した場合は、次のような問題が原因であると考えられます。後続の手順に進む前に、問題を解決する必要があります。

- データベースが起動していない。
- データベース・リスナーが起動していない。
- データベース接続が正しく構成されていない。
- ユーザー名、パスワードまたはロールが正しく入力されていない。

9. 「Add Connection」ウィンドウで、**「OK」**をクリックします。
10. Server Explorer で **HR.ORCL** 接続を開き、HR スキーマの内容を表示します。「Tables」、「Views」、「Procedures」、「Functions」、「Packages」、「Synonyms」、「Sequences」などが表示されます。

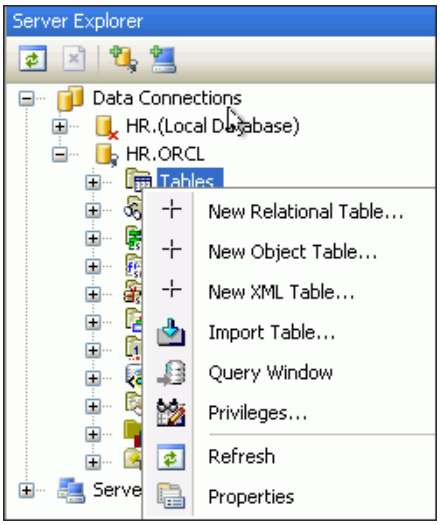


表および表の列の作成

Oracle Developer Tools には、データベース・オブジェクトを作成するためのユーザー・インタフェースが含まれています。この項では、DEPENDENTS という名前の表を作成します。

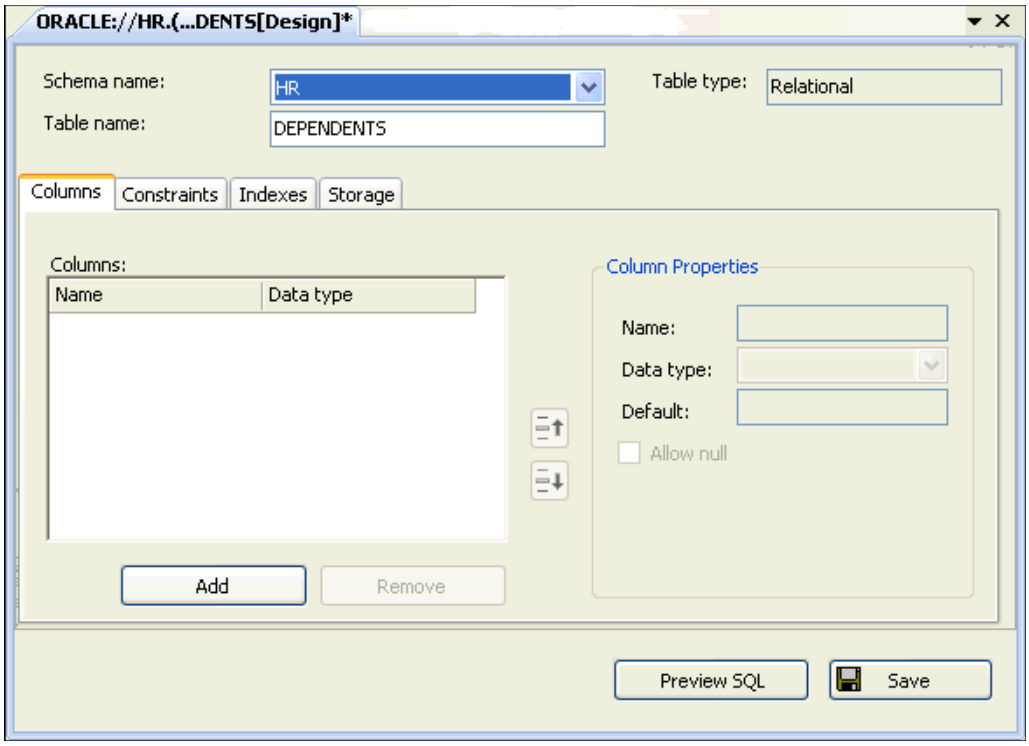
表を作成するには、次の手順を実行します。

1. Server Explorer で、**「Tables」**を右クリックして**「New Relational Table」**を選択します。



表の設計ウィンドウが表示されます。

2. 設計ビューで、「**Table name**」に DEPENDENTS と入力します。

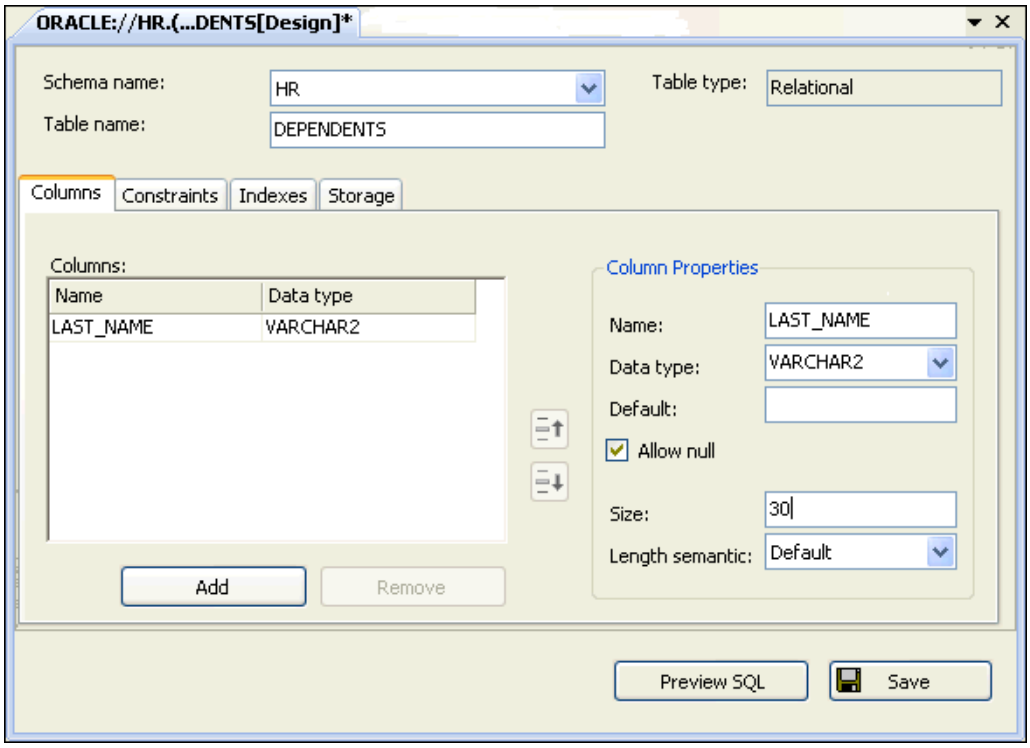


3. 「Column Properties」タブで、次に示す 6 つの列を次の方法で追加します。

   「**Add**」をクリックします。続いて、新しい列の情報を入力します。新しい列がすべて追加されるまで、繰り返し「Add」をクリックします。

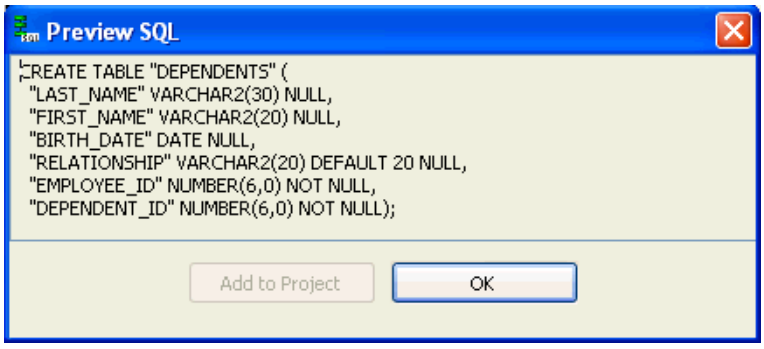
   フィールドはデータ型によって異なる場合があります。タブ全体にアクセスするには、Server Explorer や Solution Explorer などのウィンドウを閉じることが必要になる場合があります。

   - 「**Name**」に LAST_NAME、「**Data Type**」に VARCHAR2、「**Size**」に 30 を指定します。他のプロパティは、すべてデフォルト値のままにします。
   - 「**Name**」に FIRST_NAME、「**Data Type**」に VARCHAR2、「**Size**」に 20 を指定します。他のプロパティは、すべてデフォルト値のままにします。
   - 「**Name**」に BIRTH_DATE、「**Data Type**」に DATE を指定します。他のプロパティは、すべてデフォルト値のままにします。
   - 「**Name**」に RELATIONSHIP、「**Data Type**」に VARCHAR2、「**Size**」に 20 を指定します。他のプロパティは、すべてデフォルト値のままにします。
   - 「**Name**」に EMPLOYEE_ID、「**Data Type**」に NUMBER を指定し、「**Allow null**」を選択解除します。「**Precision**」に 6、「**Scale**」に 0 を入力します。
   - 「**Name**」に DEPENDENT_ID、「**Data Type**」に NUMBER を指定し、「**Allow null**」チェック・ボックスを選択解除します。「**Precision**」に 6、「**Scale**」に 0 を入力します。



4. 「**Preview SQL**」をクリックします。

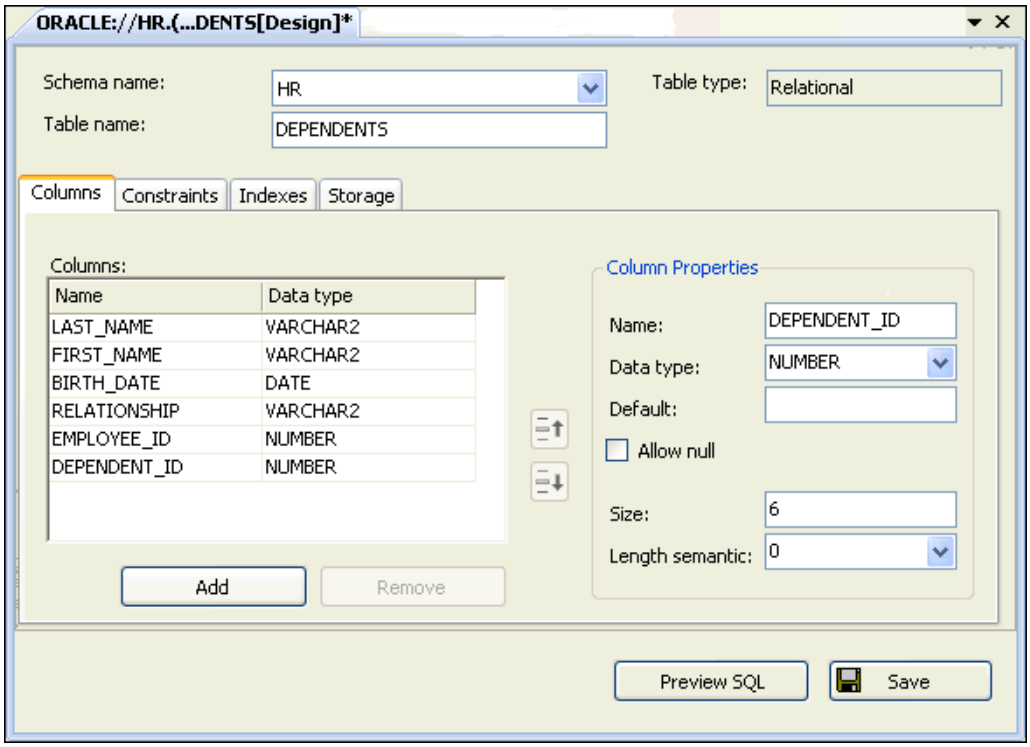
   表を作成する SQL 文が「Preview SQL」ウィンドウに次のように表示されます。



「**OK**」をクリックして「Preview SQL」ウィンドウを閉じます。

5. 表の設計ビューで「**Save**」をクリックします。

   この操作により、HR スキーマに新しい表 DEPENDENTS が作成されます。新しい表が Server Explorer にリストされます。



表の索引の作成

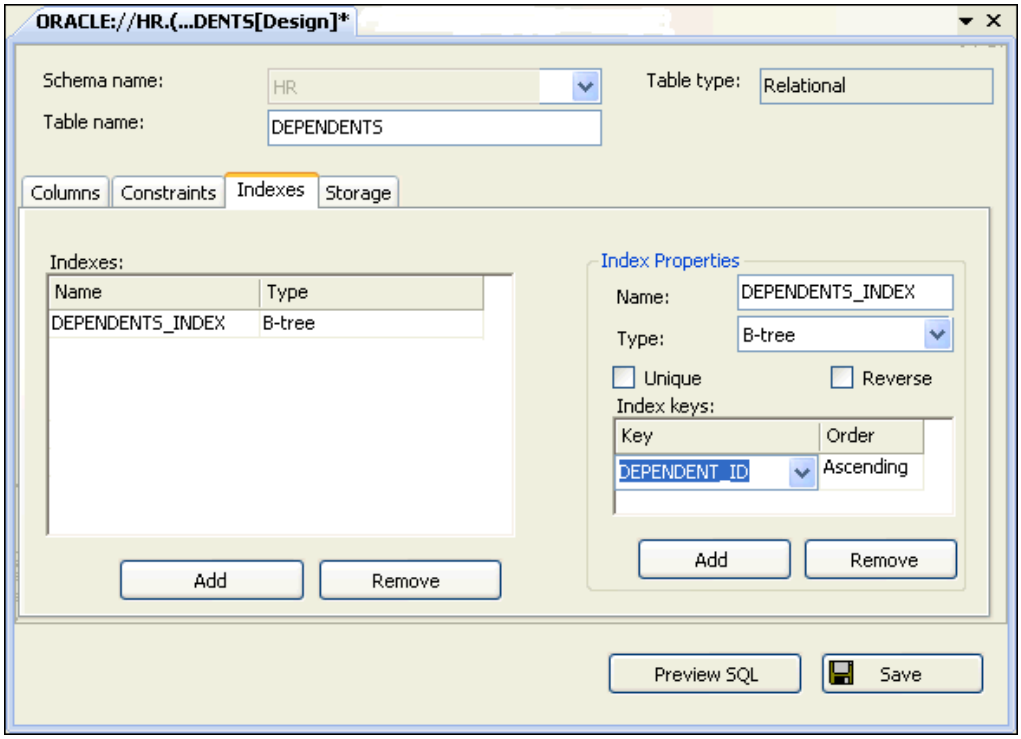
索引はオプションですが、非常に役立つリレーショナル・データベースの機能です。索引を使用すると表の行（またはレコード）にすばやくアクセスできます。この項では、DEPENDENTS 表に索引を作成します。

索引を作成するには、次の手順を実行します。

1. DEPENDENTS の表の設計ビューで「**Indexes**」タブをクリックします。
2. 「Indexes」領域の下にある「**Add**」をクリックします。

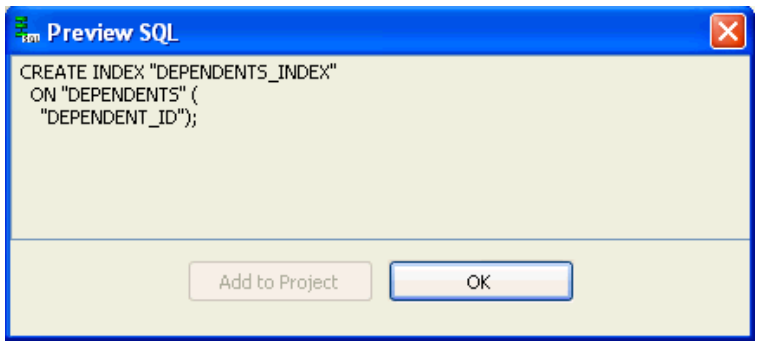
   「Index Properties」領域がアクティブになります。
3. 「Index Properties」の下（右側）にある「**Name**」に DEPENDENTS_INDEX と入力し、他のプロパティはすべてデフォルトのままにします。
4. 「Index Properties」領域の下部で、「**Add**」をクリックします。

5. 「Index keys」の下で「**Key**」列の最初のセルをクリックし、リストから「**DEPENDENT_ID**」を選択します。



6. 「**Preview SQL**」をクリックします。

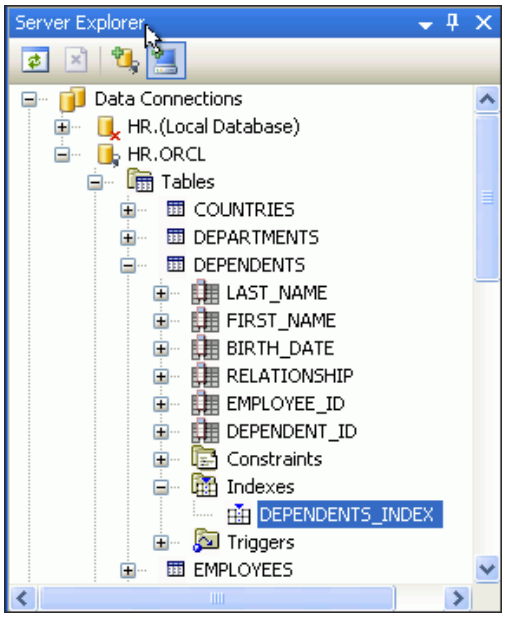
「Preview SQL」ウィンドウが表示され、索引を作成する SQL 文が表示されます。



「**OK**」をクリックして「Preview SQL」ウィンドウを閉じます。

7. 表の設計ビューで「**Save**」をクリックします。

   これで、HR スキーマの表 DEPENDENTS に新しい索引が作成されます。作成された索引を Server Explorer で確認するには、DEPENDENTS 表および関連する索引を開きます。



表の制約の追加

データベースで制約を使用すると、許容できるデータ値に対し、データ整合性の定義ルールが自動的に施行されます。また、この制約により、表の主キーや外部キーも実装されます。この項では、新しい表 DEPENDENTS にこのような制約を追加します。

外部キーおよび主キーを追加するには、次の手順を実行します。

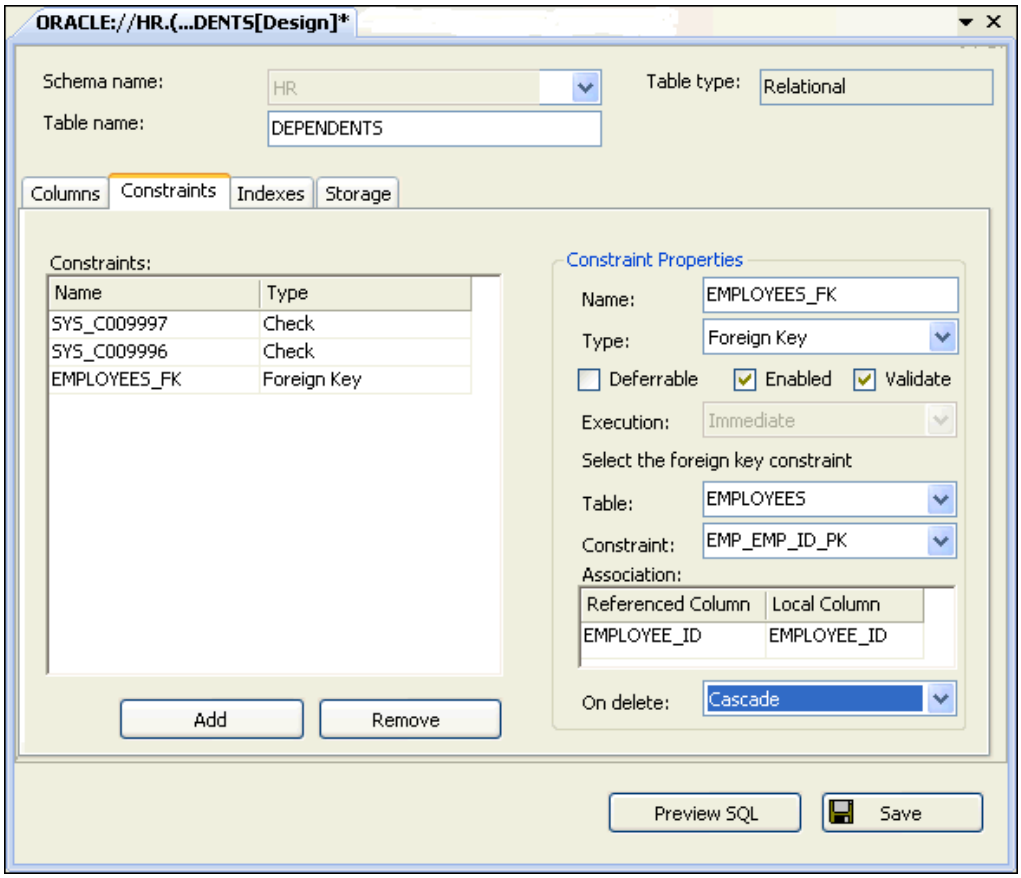
1. DEPENDENTS 表の設計ビューで、「**Constraints**」タブをクリックします。

   構成によっては、すでにデフォルトのチェック制約がリストに含まれていることがあります。

2. 「Constraints」領域の下で、次に示す制約を次の方法で追加します。

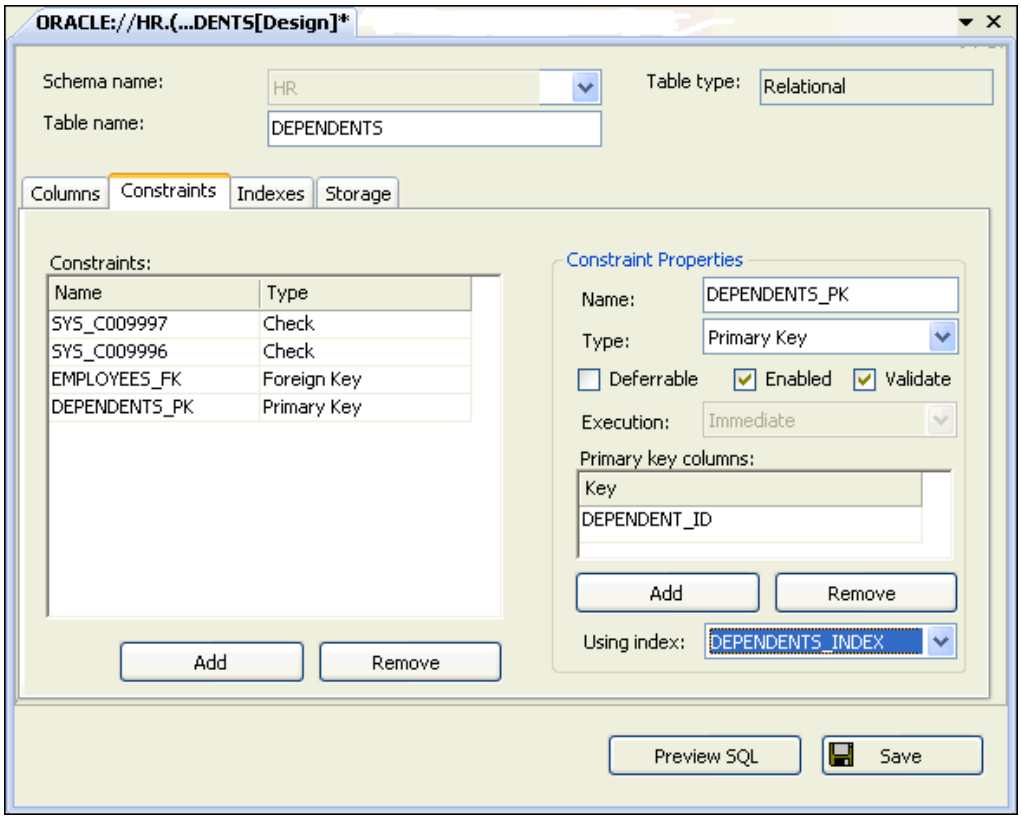
   「Constraint Properties」の下の「**Add**」をクリックします。続いて、新しい制約の情報を入力します。新しい制約がすべて追加されるまで、繰り返し「Add」をクリックします。

   - 「**Name**」に `EMPLOYEES_FK`、「**Type**」に `Foreign Key`、「**Table**」に `EMPLOYEES`、「**Constraint**」に `EMP_EMP_ID_PK` を指定します。「Association」の下で、「**Referenced Column:**」に `EMPLOYEE_ID`、「**Local Column:**」に `EMPLOYEE_ID` を選択し、「**On delete**」値に `Cascade` を設定します。他のプロパティは、すべてデフォルト値のままにします。



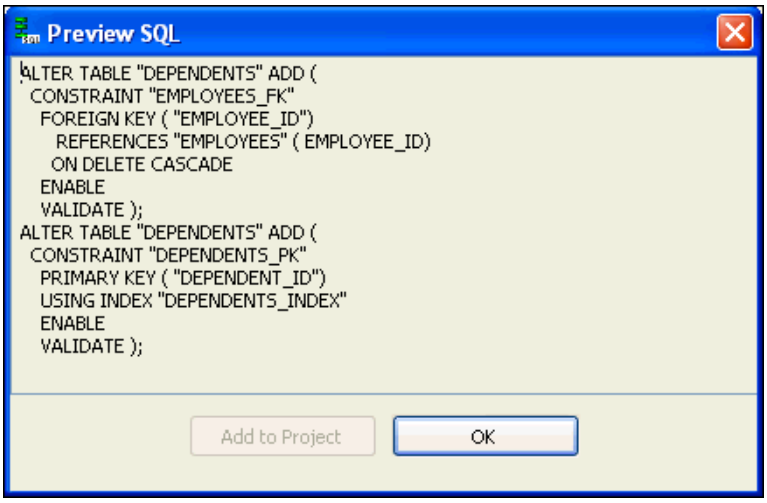
- 「**Name**」に DEPENDENTS_PK、「**Type**」に Primary Key を指定します。

  「Primary key columns」領域の下で、「**Add**」をクリックします（下にスクロールすることが必要な場合があります）。「Primary Key Columns」の下で、「**Key:**」に DEPENDENT_ID を選択し、「**Using index**」値に DEPENDENTS_INDEX を設定します。他のプロパティは、すべてデフォルト値のままにします。



3. 「**Preview SQL**」をクリックします。

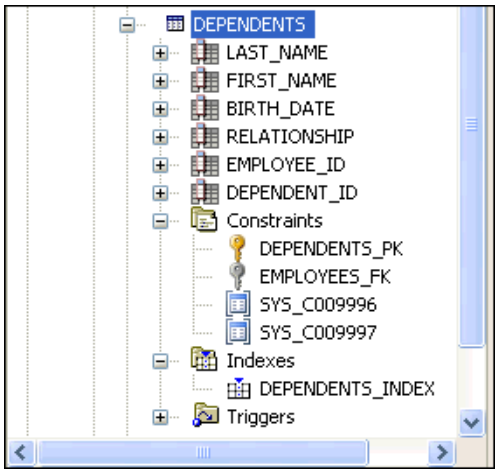
   「Preview SQL」ウィンドウに、表 DEPENDENTS の制約用に生成されたコードが表示されます。制約により、表の DEPENDENT_ID 列と EMPLOYEE_ID 列の定義が変更されるため、制約の追加には ALTER TABLE コマンドが使用されます。



「**OK**」をクリックして「Preview SQL」ウィンドウを閉じます。

4. 表の設計ビューで「**Save**」をクリックします。

この操作により、新しい 2 つの制約が HR スキーマの DEPENDENTS 表に作成されます。Server Explorer を確認するには、表 DEPENDENTS および制約の階層ツリーを開きます。

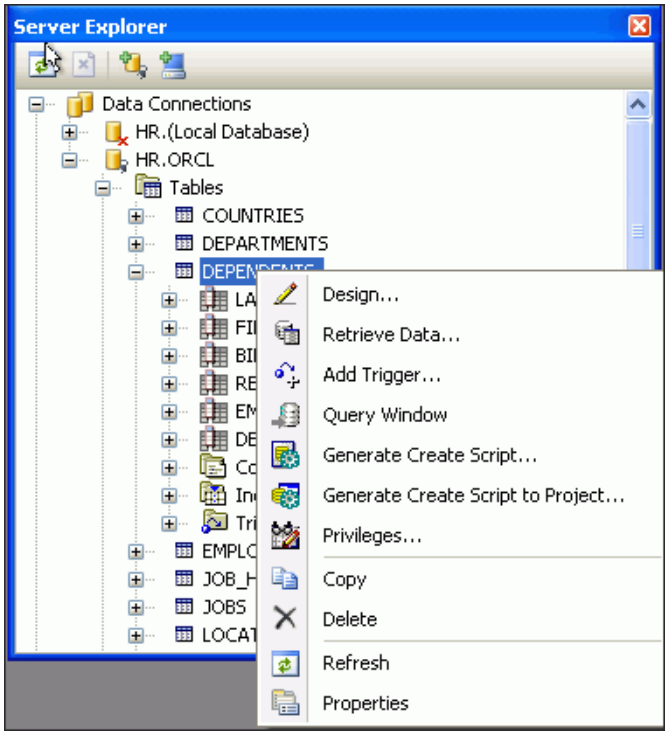


表へのデータの追加

ここで、新しい DEPENDENTS 表にデータを追加する必要があります。

表にデータを移入するには、次の手順を実行します。

1. Server Explorer で、DEPENDENTS 表を右クリックして「**Retrieve Data**」を選択します。



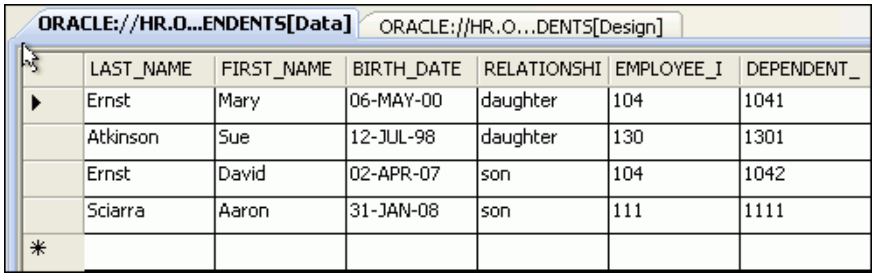
DEPENDENTS の表グリッドが設計ビューに表示されます。

2. 表 5-1 に示す 4 つのレコードを、この表グリッドに追加します。

表 5-1 DEPENDENTS 表の新しいデータ

LAST_NAME	FIRST_NAME	BIRTH_DATE	RELATIONSHIP	EMPLOYEE_ID	DEPENDENT_ID
Ernst	Mary	06-MAY-2000	daughter	104	1041
Atkinson	Sue	12-JUL-1998	daughter	130	1301
Ernst	David	02-APR-2007	son	104	1042
Sciarra	Aaron	31-JAN-2008	son	111	1111

グリッドは次のようになります。



行を移動すると、データは自動的に保存されます。

データを表示および更新するためのコードの自動生成

DEPENDENTS 表の内容を確認するために、表の単純な問合せを使用するフォームを作成します。この項では、Visual Studio 統合開発環境（IDE）を使用して、操作に対応するコードを自動生成します。

新しいデータソースを作成するには、次の手順を実行します。

1. 3-2 ページの「新しいプロジェクトの作成」の説明に従って、新しいプロジェクトを開始します。新しいプロジェクトには、次に示す名前を付けます。

   Visual C#:

   `HR_ODT_CS`

   Visual Basic:

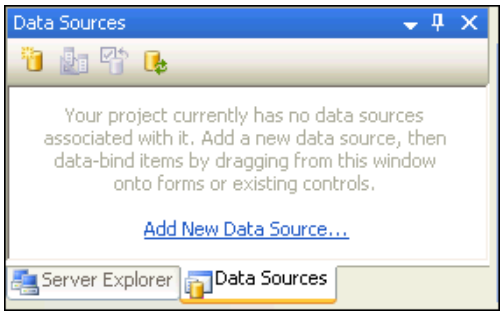
   `HR_ODT_VB`

2. 「**Create Directory for Solution**」を選択します。「**OK**」をクリックします。
3. 設計ビューが表示されていない場合は、Form1 の設計ビューに切り替えます。

   注意 : すべてのアプリケーションは Form1 から開始しますが、これまでの章で作成したアプリケーションとの関係はありません。
4. 「Server Explorer」ウィンドウをクリックし、「Show Data Sources」ウィンドウを有効にします。

5. Visual Studio の「**Data**」メニューから「**Show Data Sources**」を選択します。

   「Data Source」ウィンドウが表示されます。

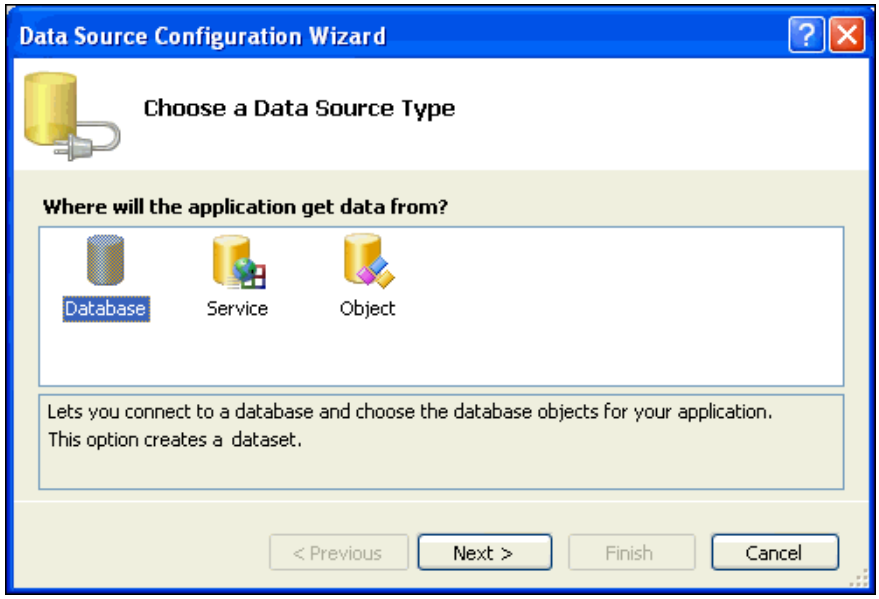


6. 「Data Sources」ウィンドウで、「**Add New Data Source**」をクリックします。

   「Data Source Configuration Wizard」が開きます。

7. 「Data Source Configuration Wizard」の「Choose a Data Source Type」で「**Database**」を選択します。

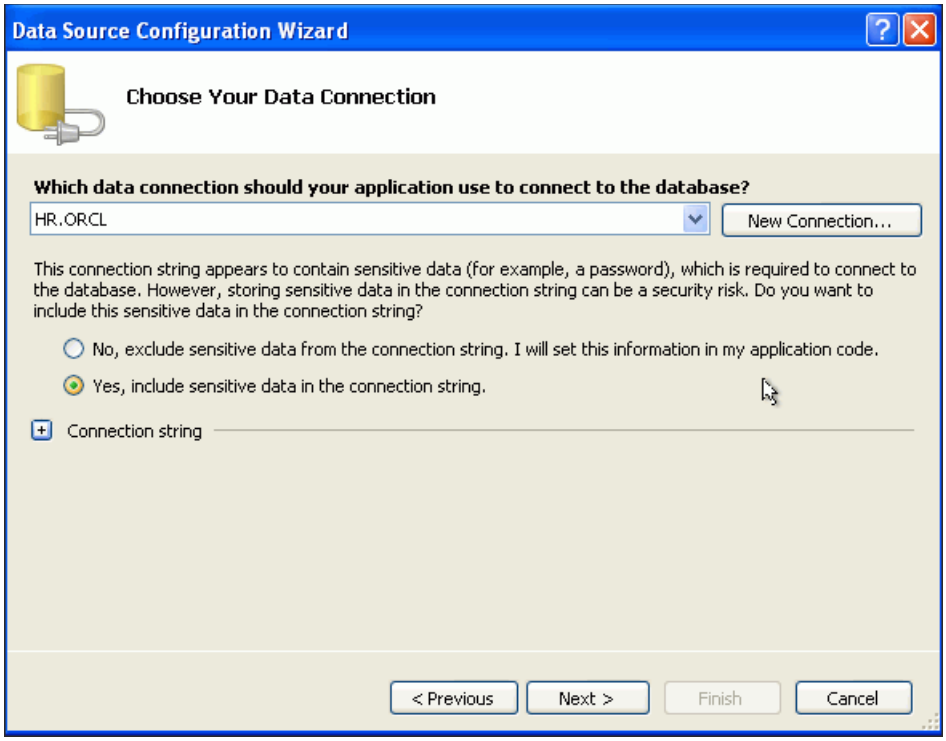
   「**Next**」をクリックします。



8. 「Choose Your Data Connection」で、「**HR.ORCL**」または「**HR.(Local Database)**」を選択します。この例では、HR.ORCL を使用します。

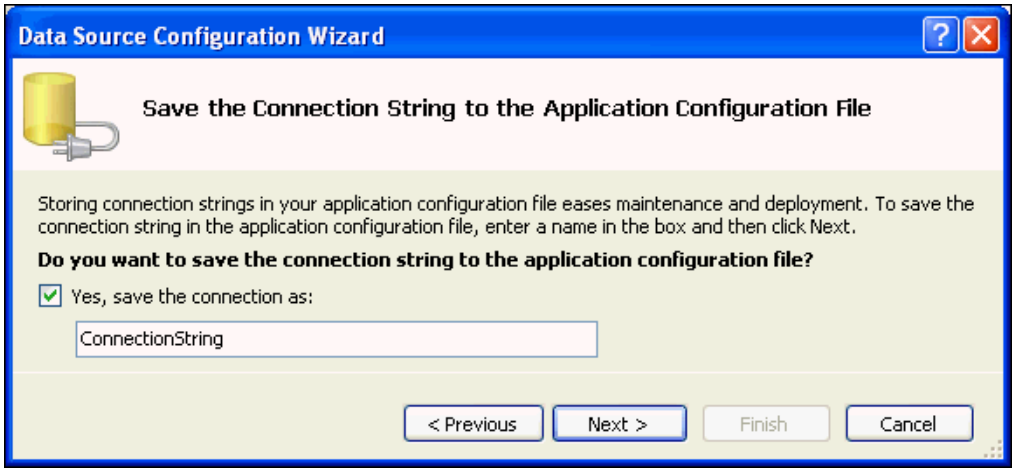
   「**Yes, include sensitive data in the connection string**」を選択します。

   「**Next**」をクリックします。



9. 「Save the Connection String to the Application Configuration File」で、「**Yes, save the connection as:**」に `ConnectionString` を選択します。

   「**Next**」をクリックします。

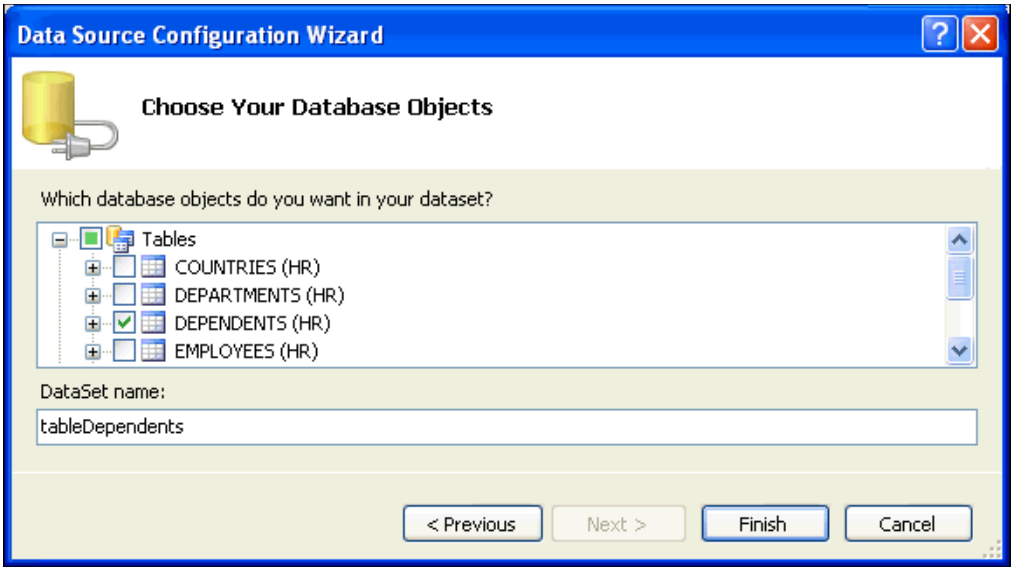


10. 「Choose Your Database Objects」で、「**Tables**」を開きます。

    「**DEPENDENTS(HR)**」表を選択します。

    「**DataSet name**」を **tableDependents** に変更します。

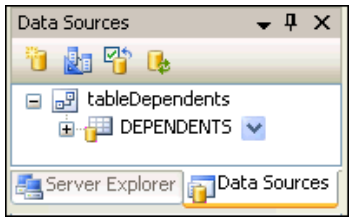
    「**Finish**」をクリックします。



参照： DataSet クラスの詳細は、4-8 ページの「Oracle Data Provider for .NET での DataSet クラスの使用」を参照してください。

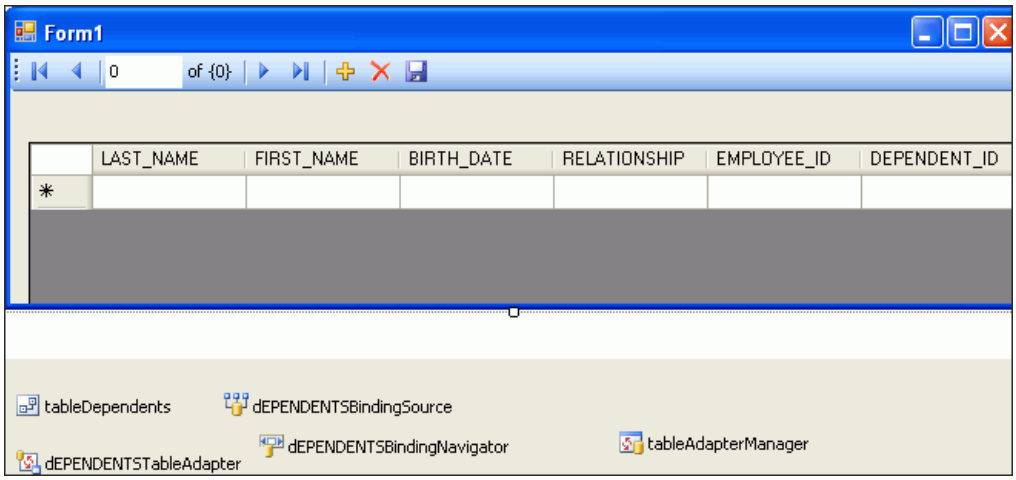
ドラッグ・アンド・ドロップでコードを自動生成するには、次の手順を実行します。

1. Form1 の設計ビューに切り替えます。
2. 「Data Sources」ウィンドウで、「**tableDependents**」を開きます。



3. **DEPENDENTS** 表を選択し、Form1 にドラッグします。

   フォームと表グリッドの両方のサイズ調整が必要になる場合があります。



   表グリッド（レコード・ナビゲーション要素を含む）の他に、次のコンポーネントがプロジェクトの設計ビューに追加されます。これらのオブジェクトは、Form1 に自動生成されるコードを表しています。

   Visual C#:

   `tableDependents`、`dEPENDENTSBindingSource`、`dEPENDENTSTableAdapter`、`tableAdapterManager` および `dEPENDENTSBindingNavigator`

   Visual Basic:

   `TableDependents`、`DEPENDENTSBindingSource`、`DEPENDENTSTableAdapter`、`TableAdapterManager` および `DEPENDENTSBiningNavigator`

4. Form1 の上部付近にある「**Save**」アイコン（フロッピー・ディスク）をダブルクリックします。

   これにより、Form1 の「Save」アイコンのコード・ウィンドウが開きます。

5. プライベート・メソッド *xxx*`SaveItem_Click()` で、`try...catch` ブロックに既存のコードをカプセル化します。この自動生成されたメソッドの Visual C# および Visual Basic の完全な名前は、次に示すコードを参照してください。

   また、`MessageBox.show()` コールを Try セクションおよび Catch セクションの両方に追加します。更新したメソッド・コードを次に示します。新しいコードまたは変更されたコードは太字で表しています。

   Visual C#:

```
private void dEPENDENTSBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
 try
 {
  this.Validate();
  this.dEPENDENTSBindingSource.EndEdit();
  this.tableAdapterManager.UpdateAll(this.tableDependents);

  MessageBox.Show("Update successful");
 }
 catch (System.Exception ex)
 {
  MessageBox.Show("Update failed: "+ ex.Message.ToString());
 }
}
```

Visual Basic:

```
Private Sub DEPENDENTSBindingNavigatorSaveItem_Click(
 ByVal sender As System.Object, ByVal e As System.EventArgs)
 Handles DEPENDENTSBindingNavigatorSaveItem.Click

 Try
  Me.Validate()
  Me.DEPENDENTSBindingSource.EndEdit()
  Me.TableAdapterManager.UpdateAll(Me.TableDependents)
  MessageBox.Show("Update successful")

 Catch ex As Exception
  MessageBox.Show("Update failed: " + ex.Message.ToString())

 End Try

End Sub
```

6. アプリケーションをコンパイルして実行するには、3-13 ページの「アプリケーションのコンパイルと実行」の手順に従います。

次の方法で、新しいアプリケーションをテストできます。フロッピー・ディスク・アイコンは、「Save」コマンドを表しています。

アプリケーションをテストするには、次の手順を実行します。

1. Mary Ernst の「DEPENDENT_ID」の値を 1110 に変更し、**「Save」**アイコンをクリックします。メッセージ・ボックス Update successful が表示されます。**「OK」**をクリックしてメッセージ・ボックスを閉じます。
2. David Ernst の「EMPLOYEE_ID」の値を 99999 に変更し、**「Save」**アイコンをクリックします。「更新に失敗しました。: ORA-02291: 整合性制約 (HR.EMPLOYEES_FK) に違反しました - 親キーがありません」というメッセージが表示されます。**「OK」**をクリックしてメッセージ・ボックスを閉じます。

6

PL/SQL ストアド・プロシージャおよび REF CURSOR の使用

この章の内容は次のとおりです。

PL/SQL ストアド・プロシージャの概要

ストアド・プロシージャとは、ある操作を実行するために設計された一連の PL/SQL 文を 1 つにまとめて名前を付けたものです。ストアド・プロシージャはデータベース内に格納されます。これは、クライアント・アプリケーションがデータベース・オブジェクトと直接対話できるようにするものではなく、データベースのプログラミング・インタフェースを定義するものです。一般的にストアド・プロシージャを使用するのは、データを検証する場合や、複数の SQL 問合せを組み合せた大規模で複雑な処理の指示をカプセル化する場合です。

ストアド・ファンクションは、戻り値パラメータを 1 つとります。ファンクションとは異なり、プロシージャは値を戻す場合と戻さない場合があります。

PL/SQL パッケージとパッケージ本体の概要

PL/SQL パッケージには、関連する項目が単一の論理エンティティとして格納されます。パッケージは、次の 2 つの部分で構成されます。

- **パッケージ仕様部**では、パッケージに含まれるものを定義します。これは、C++ などの言語のヘッダー・ファイルに似ています。仕様部では、すべてのパブリック項目を定義します。仕様部はパッケージの公開インタフェースです。
- **パッケージ本体**には、仕様部で定義したプロシージャおよびファンクションのコードと、仕様部で宣言していないプライベート・プロシージャおよびファンクションのコードが含まれます。プライベート・コードはパッケージ本体内でのみ確認できます。

パッケージ仕様部とパッケージ本体は、データ・ディクショナリに別々のオブジェクトとして格納され、`user_source` ビューで確認できます。仕様部は `PACKAGE` 型として格納され、本体は `PACKAGE BODY` 型として格納されます。

一連のパブリック定数を宣言する場合と同様、本体のない仕様部を保持することはできますが、仕様部のない本体を保持することはできません。

REF CURSOR の概要

`REF CURSOR` を使用することは、Oracle Database からの問合せ結果をクライアント・アプリケーションに戻す最も強力かつ柔軟で、拡張性のある方法の 1 つです。

`REF CURSOR` は PL/SQL データ型であり、この値はデータベース上の問合せ作業領域のメモリー・アドレスです。つまり、`REF CURSOR` は、データベース上にある結果セットへのポインタまたはハンドルとなります。`REF CURSOR` は、`OracleRefCursor` ODP.NET クラスを使用して表現します。

`REF CURSOR` には次の特性があります。

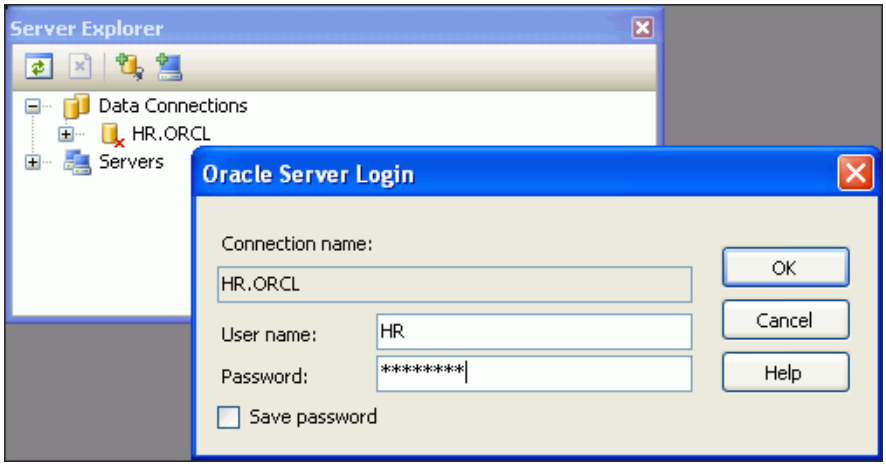
- `REF CURSOR` は、データベース上のメモリー・アドレスを参照します。そのため、`REF CURSOR` にアクセスするには、`REF CURSOR` の存続期間中、クライアントがデータベースに接続されている必要があります。
- `REF CURSOR` により、追加のデータベース・ラウンドトリップが発生します。`REF CURSOR` がクライアントに戻されても、クライアントが `REF CURSOR` をオープンしてデータをリクエストするまで、実際のデータは戻されません。ユーザーが REF CURSOR の読取りを試行するまで、データは取得されないことに注意してください。
- `REF CURSOR` は更新できません。`REF CURSOR` で表される結果セットは読取り専用です。`REF CURSOR` を使用してデータベースを更新することはできません。
- `REF CURSOR` は後方にスクロールできません。`REF CURSOR` で表される結果セットは、前進専用で順次アクセスされます。結果セット内のレコードをランダムにポイントするために `REF CURSOR` 内にレコード・ポインタを配置することはできません。
- `REF CURSOR` は PL/SQL データ型です。PL/SQL コード・ブロック内で `REF CURSOR` を作成して戻すことができます。

REF CURSOR を使用する PL/SQL ストアド・プロシージャの作成

この項では、PL/SQL ストアド・プロシージャの作成方法を説明します。

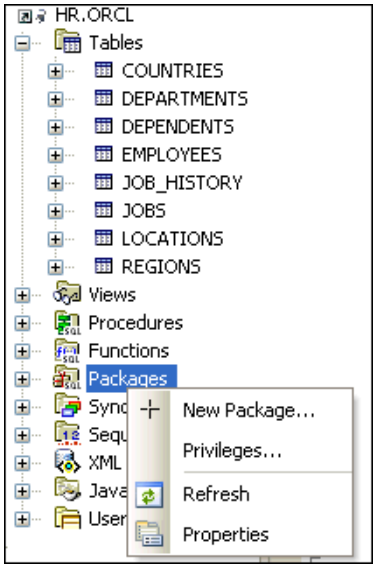
ストアド・プロシージャを作成するには、次の手順を実行します。

1. Server Explorer を開いて HR をダブルクリックし、5-2 ページの「Oracle Database への接続」で作成した HR スキーマへの接続をオープンします。



以前にパスワードを保存しなかった場合は、「Oracle Server Login」が開き、パスワードを入力できます。パスワードを以前に保存している場合は、すぐに接続がオープンします。

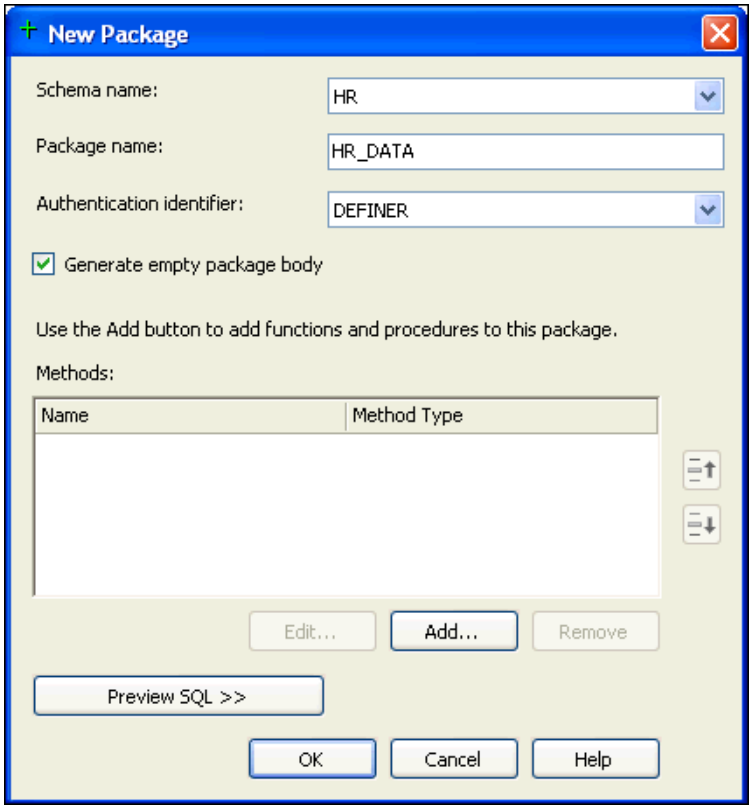
2. Server Explorer で、**「Packages」**を右クリックして**「New Package」**を選択します。



「New Package」ウィンドウが表示されます。

3. 「New Package」ウィンドウで、**「Package Name」**を `HR_DATA` に変更します。

4. 「Methods」領域の下の「**Add**」をクリックします。



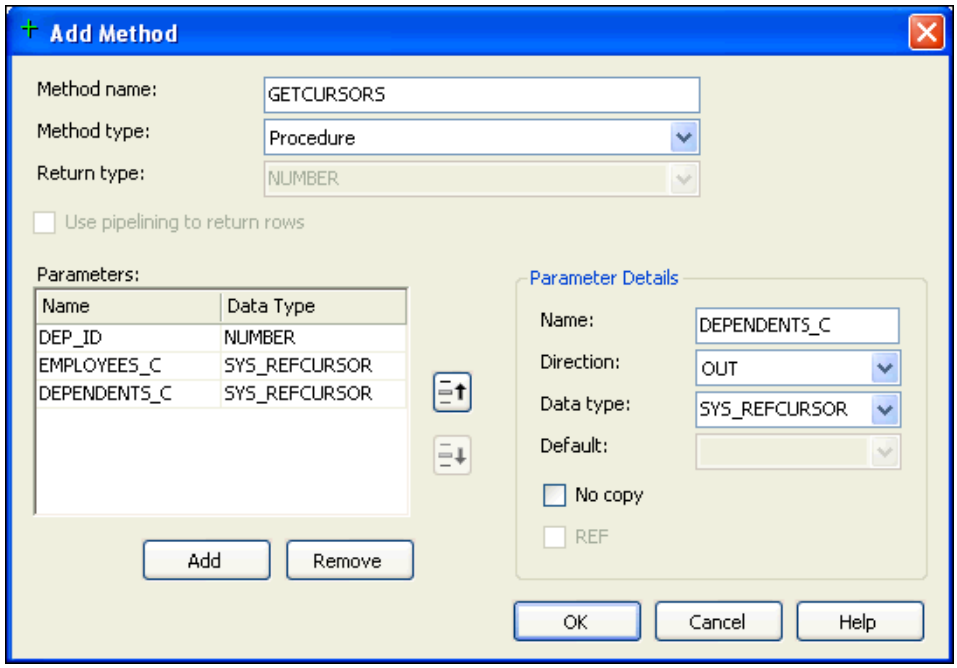
「Add Method」ウィンドウが表示されます。

5. 「Add Method」ウィンドウで、「**Method Name**」に `GETCURSORS` と入力し、「**Method Type**」を `Procedure` に変更します。
6. 「Parameters」の下の「**Add**」をクリックします。

   これにより、パラメータを追加するプロセスが開始されます。

   右側の「Parameter Details」グループで、次の 3 つのパラメータを入力します。「**Add**」をクリックしてから、必要なパラメータを 1 つずつ追加します。

   - 「**Name:**」に `DEP_ID` と入力し、「**Direction:**」は「`IN`」を選択します。「**Data Type:**」には「`NUMBER`」を選択します。
   - 「**Name:**」に `EMPLOYEES_C` と入力し、「**Direction:**」は「`OUT`」を選択します。「**Data Type:**」には「`SYS_REFCURSOR`」を選択します。
   - 「**Name:**」に `DEPENDENTS_C` と入力し、「**Direction:**」は「`OUT`」を選択します。「**Data Type:**」には「`SYS_REFCURSOR`」を選択します。



7. パラメータの追加が終了したら、**「OK」**をクリックします。

   「New Package」ウィンドウが再度表示されます。

8. 「New Package」ウィンドウで**「Preview SQL」**をクリックし、作成された SQL コードを確認します。

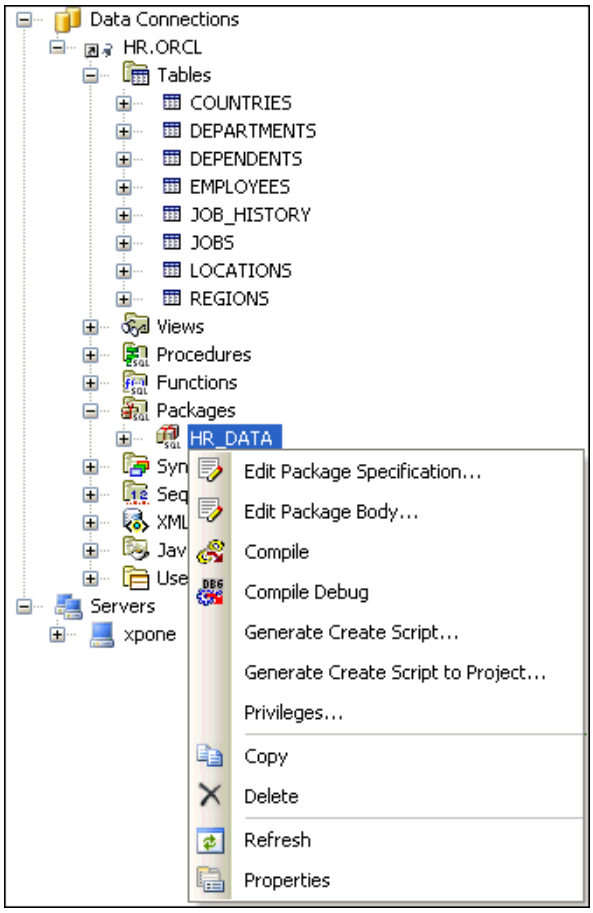
   次のようなコードを含む「Preview SQL」ウィンドウが表示されます。このコードは、コメントの大部分を削除して短縮したものです。

```
CREATE PACKAGE "HR"."HR_DATA" IS

  -- Declare types, variables, constants, exceptions, cursors,
  -- and subprograms that can be referenced from outside the package.

  PROCEDURE "GETCURSORS" (
    "DEP_ID" IN NUMBER,
    "EMPLOYEES_C" OUT SYS_REFCURSOR,
    "DEPENDENTS_C" OUT SYS_REFCURSOR);

END "HR_DATA";

CREATE PACKAGE BODY "HR"."HR_DATA" IS

  -- Implement subprograms, initialize variables declared in package
  -- specification.

  -- Make private declarations of types and items, that are not accessible
  -- outside the package

  PROCEDURE "GETCURSORS" (
    "DEP_ID" IN NUMBER,
    "EMPLOYEES_C" OUT SYS_REFCURSOR,
    "DEPENDENTS_C" OUT SYS_REFCURSOR) IS

  -- Declare constants and variables in this section.

     BEGIN -- executable part starts here
```

```
    NULL;

  -- EXCEPTION -- exception-handling part starts here

    END "GETCURSORS";

END "HR_DATA";
```

9. 「**OK**」をクリックして「Preview SQL」ウィンドウを閉じます。
10. 「New Package」ウィンドウで「**OK**」をクリックし、新しいパッケージを保存します。

    新しいパッケージ HR_DATA が Server Explorer に表示されます。
11. Server Explorer で、パッケージ HR_DATA を右クリックして「**Edit Package Body**」を選択します。



パッケージのコードが表示されます。

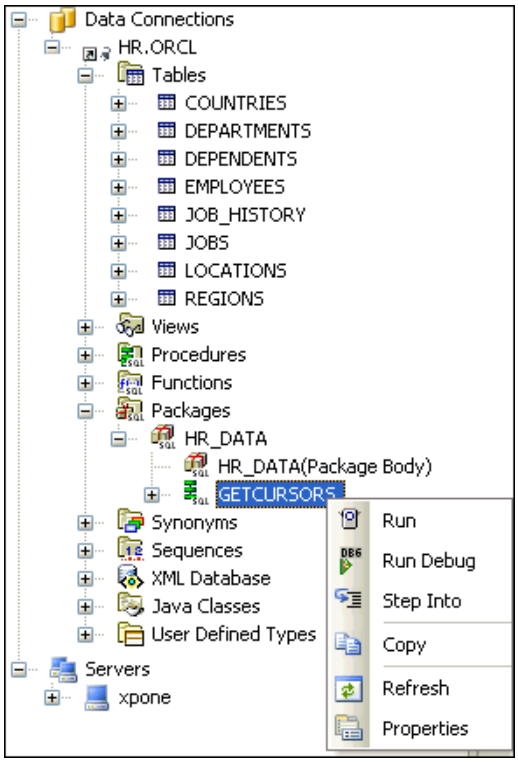
12. GETCURSORS プロシージャの本体までスクロールし、BEGIN の後にある行 NULL; を次のコードに置き換えます。

```
OPEN EMPLOYEES_C FOR SELECT * FROM EMPLOYEES
   WHERE DEP_ID=DEPARTMENT_ID;
OPEN DEPENDENTS_C FOR SELECT * FROM DEPENDENTS;
```

13. パッケージへの変更を保存します。

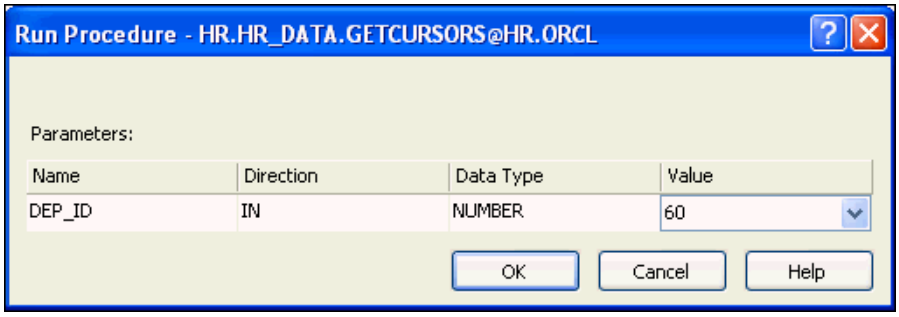
14. ストアド・プロシージャを実行するには、Server Explorer で HR_DATA パッケージを開きます。

    GETCURSORS メソッドを右クリックして、「**Run**」を選択します。



    「Run Procedure」ウィンドウが表示されます。

15. 「Run Procedur」ウィンドウで、DEP_ID の「**Value**」に 60 を入力します。



16. 「**OK**」をクリックします。

    「Output」ウィンドウが表示され、正しく実行されたことが示されます。

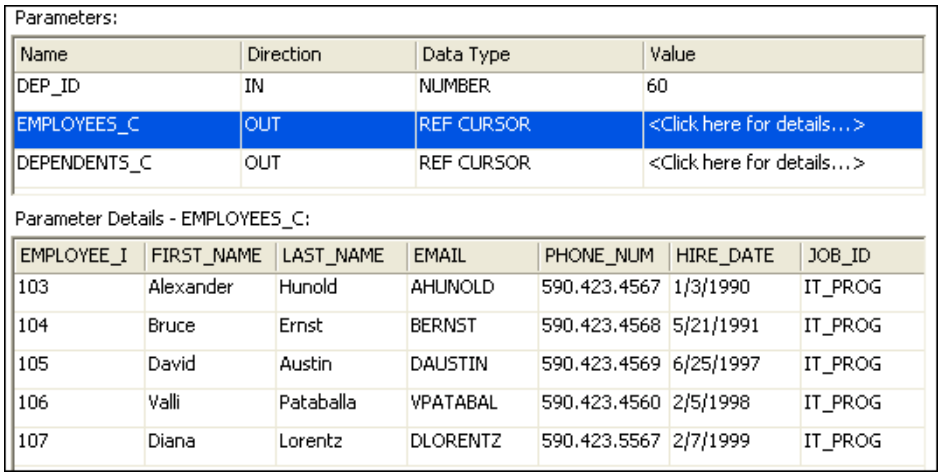
    結果ウィンドウに次のメッセージが表示されます。

```
Procedure <HR.HR_DATA.GETCURSORS@hr.database> was run successfully.
```

    このメッセージの下の（DEP_ID とともに表示される）2 つの出力パラメータ（EMPLOYEES_C および DEPENDENTS_C）を確認します。

17. EMPLOYEES_C の「**Value**」列のエントリを選択します。

「Parameter Details」領域が表示され、部門 60 の従業員が表示されます。DEP_ID の値は 60 です。



Parameters:

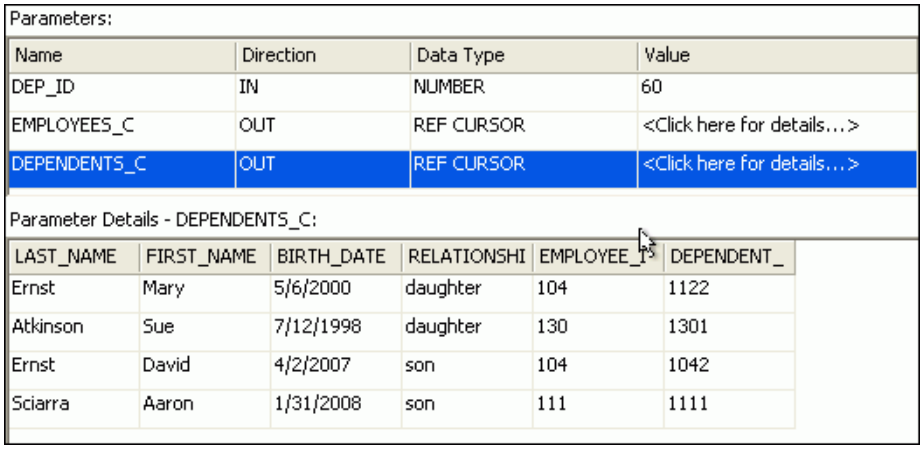
Name	Direction	Data Type	Value
DEP_ID	IN	NUMBER	60
EMPLOYEES_C	OUT	REF CURSOR	<Click here for details...>
DEPENDENTS_C	OUT	REF CURSOR	<Click here for details...>

Parameter Details - EMPLOYEES_C:

EMPLOYEE_I	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUM	HIRE_DATE	JOB_ID
103	Alexander	Hunold	AHUNOLD	590.423.4567	1/3/1990	IT_PROG
104	Bruce	Ernst	BERNST	590.423.4568	5/21/1991	IT_PROG
105	David	Austin	DAUSTIN	590.423.4569	6/25/1997	IT_PROG
106	Valli	Pataballa	VPATABAL	590.423.4560	2/5/1998	IT_PROG
107	Diana	Lorentz	DLORENTZ	590.423.5567	2/7/1999	IT_PROG

18. DEPENDENTS_C の「**Value**」列のエントリを選択します。

「Parameter Details」領域が表示され、DEPENDENTS_C の値が表示されます。



Parameters:

Name	Direction	Data Type	Value
DEP_ID	IN	NUMBER	60
EMPLOYEES_C	OUT	REF CURSOR	<Click here for details...>
DEPENDENTS_C	OUT	REF CURSOR	<Click here for details...>

Parameter Details - DEPENDENTS_C:

LAST_NAME	FIRST_NAME	BIRTH_DATE	RELATIONSHI	EMPLOYEE_I	DEPENDENT_
Ernst	Mary	5/6/2000	daughter	104	1122
Atkinson	Sue	7/12/1998	daughter	130	1301
Ernst	David	4/2/2007	son	104	1042
Sciarra	Aaron	1/31/2008	son	111	1111

ストアド・プロシージャを実行するための ODP.NET アプリケーションの変更

この項では、Oracle Data Provider for .NET アプリケーションを変更して PL/SQL ストアド・プロシージャを実行できるようにする方法を、GETCURSORS ストアド・プロシージャを例に説明します。

ストアド・プロシージャを実行できるようにアプリケーションを変更するには、次の手順を実行します。

1. アプリケーション HR_Connect_CS または HR_Connect_VB を開きます。
2. 付録 B「フォームのコピー」の手順に従って、第 4 章の最後で完成させた Form3.*xx* のコピーを作成し、Form4.xx という名前を付けます。

3. Form1 を選択し、コード・ビューに切り替えます。
4. connect_Click() メソッドの Try ブロックで、コマンドを割り当てる 2 つの行（cmd = New OracleCommand... で始まる行）を、次に示すコードと置き換えます。

Visual C#:

```
cmd = new OracleCommand("HR_DATA.GETCURSORS", conn);
cmd.CommandType = CommandType.StoredProcedure;
```

Visual Basic:

```
cmd = new OracleCommand("HR_DATA.GETCURSORS", conn)
cmd.CommandType = CommandType.StoredProcedure
```

5. 手順 4 で追加したコードの下に、GETCURSORS ストアド・プロシージャの 3 つのパラメータの定義とバインドを、それぞれ dep_id、employees_c および dependents_c という名前の OracleParameter オブジェクトとして追加します。

Visual C#:

```
OracleParameter dep_id = new OracleParameter();
dep_id.OracleDbType = OracleDbType.Decimal;
dep_id.Direction = ParameterDirection.Input;
dep_id.Value = 60;
cmd.Parameters.Add(dep_id);

OracleParameter employees_c = new OracleParameter();
employees_c.OracleDbType = OracleDbType.RefCursor;
employees_c.Direction = ParameterDirection.Output;
cmd.Parameters.Add(employees_c);

OracleParameter dependents_c = new OracleParameter();
dependents_c.OracleDbType = OracleDbType.RefCursor;
dependents_c.Direction = ParameterDirection.Output;
cmd.Parameters.Add(dependents_c);
```

Visual Basic:

```
Dim dep_id As OracleParameter = New OracleParameter
dep_id.OracleDbType = OracleDbType.Decimal
dep_id.Direction = ParameterDirection.Input
dep_id.Value = 60
cmd.Parameters.Add(dep_id)

Dim employees_c As OracleParameter = New OracleParameter
employees_c.OracleDbType = OracleDbType.RefCursor
employees_c.Direction = ParameterDirection.Output
cmd.Parameters.Add(employees_c)

Dim dependents_c As OracleParameter = New OracleParameter
dependents_c.OracleDbType = OracleDbType.RefCursor
dependents_c.Direction = ParameterDirection.Output
cmd.Parameters.Add(dependents_c)
```

6. アプリケーションをビルドします。

ODP.NET アプリケーションによる PL/SQL ストアド・プロシージャの実行

この項では、GETCURSORS ストアド・プロシージャなどの PL/SQL ストアド・プロシージャを、ODP アプリケーションから実行する方法を説明します。

ストアド・プロシージャを実行するには、次の手順を実行します。

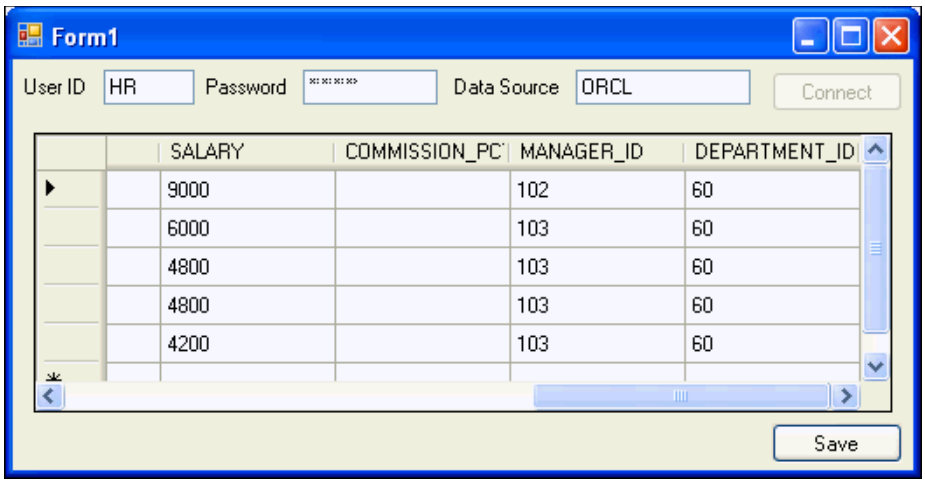
1. アプリケーションを実行します。

   「Form1」ウィンドウが表示されます。

2. 「Form1」ウィンドウで接続情報を入力し、「**Connect**」をクリックします。

3. DataGrid オブジェクトで、水平方向にスクロールして最後の列 DEPARTMENT_ID に含まれる値が 60 のみであることを確認します。

   DataGrid には、ストアド・プロシージャの最初の結果セットが含まれており、これは EMPLOYEES 表の問合せと一致することに注意してください。



4. アプリケーションを閉じます。

7

Oracle Database での ASP.NET の使用

この章の内容は次のとおりです。

概要 : Oracle Developer Tools での ASP.NET アプリケーションの作成

Oracle を Microsoft ASP.NET と直接統合する方法は多くあります。

- Oracle Developer Tools for Visual Studio を使用すると、データ駆動型の Web サイトを簡単に設計できます。
- ODP.NET を使用すると、ASP.NET データにアクセスできます。
- Oracle Providers for ASP.NET を Microsoft ASP.NET 制御およびサービスと直接統合することで、Web サイトの状態管理機能を使用できます。

このチュートリアルでは、Oracle Developer Tools を使用したデータ駆動型 Web アプリケーションの作成方法、Oracle Providers for ASP.NET を使用した単純な方法でのデータ駆動型 Web アプリケーションへのセキュリティの追加など、いくつかの機能を示します。

まず、ツールを使用して、Oracle Database からデータ・グリッドに従業員データを取得する Web アプリケーションを作成します。次に、ログイン・コントロールを追加し、認可された Web ユーザーにのみこの従業員情報へのアクセスを許可することでアプリケーションを保護します。最後に、Oracle Providers for ASP.NET を使用して認可された Web ユーザーを作成します。これらの Web ユーザーは、このアプリケーションでの認証用に Oracle Database 内に格納されます。

このチュートリアルを開始する前に

Oracle Developer Tools で ASP.NET アプリケーションを作成する前に、後続の項で説明する設定を実行する必要がある場合があります。

- 5-2 ページの「Oracle Database への接続」
- Web サイト認証（7-12 ページの「認証用の Web サイトの有効化」を参照）を使用する予定がある場合は、次の設定を実行する必要があります。
  - 2-11 ページの「ユーザーの作成および権限の付与」
  - 2-15 ページの「Oracle Providers for ASP.NET のすべての構成」
  - 2-19 ページの「接続文字列の設定」

Web サイトの作成およびデータベースへの接続

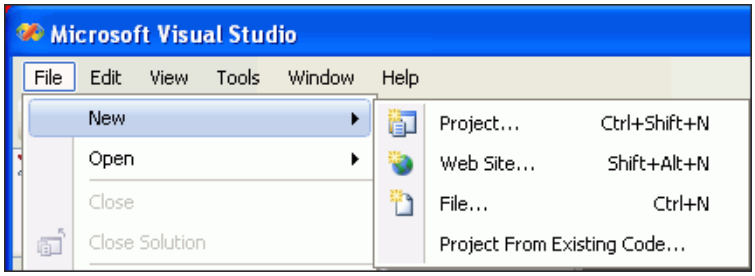
この項では、Oracle Database からデータを取得する ASP.NET Web サイトの作成方法について説明します。Web サイトの ASP.NET GridView にデータが表示されるため、ユーザーは結果をページごとに確認できます。

この項の内容は次のとおりです。

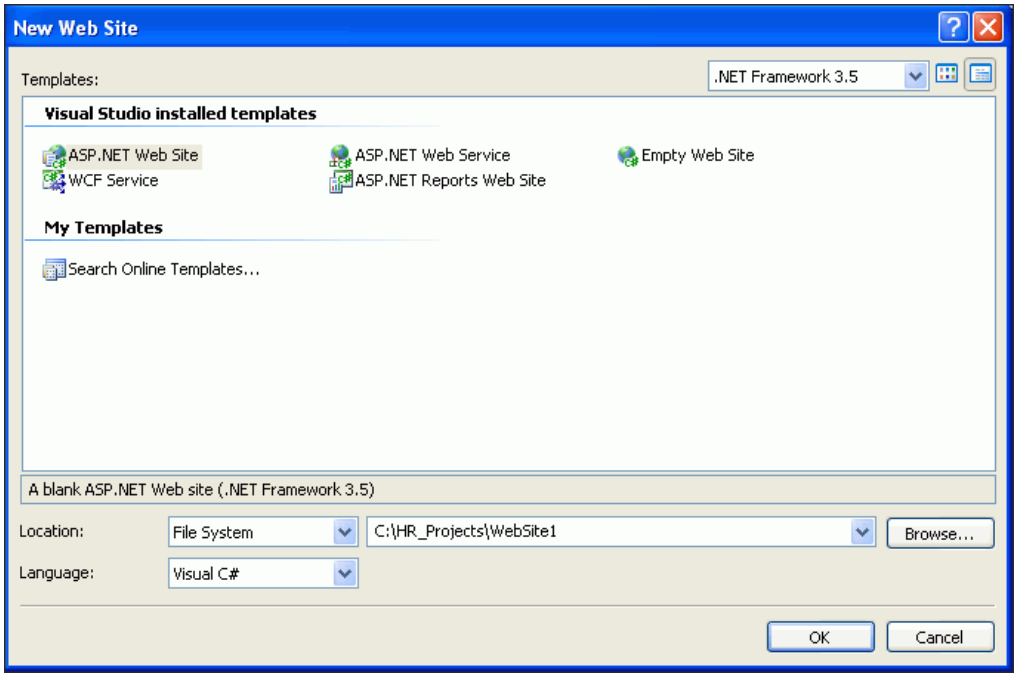
ASP.NET Web サイトの作成

グリッド付きの ASP.NET Web サイトを作成するには、次の手順を実行します。

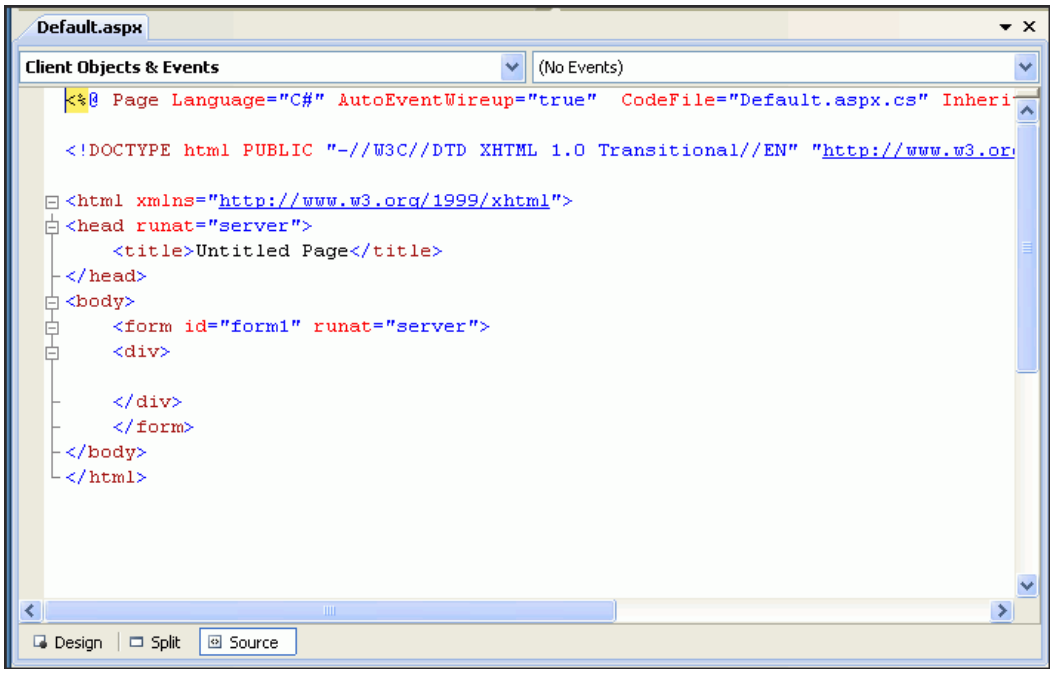
1. Visual Studio を起動します。
2. 「File」メニューから、**「New」**→**「Web Site...」**を選択します。



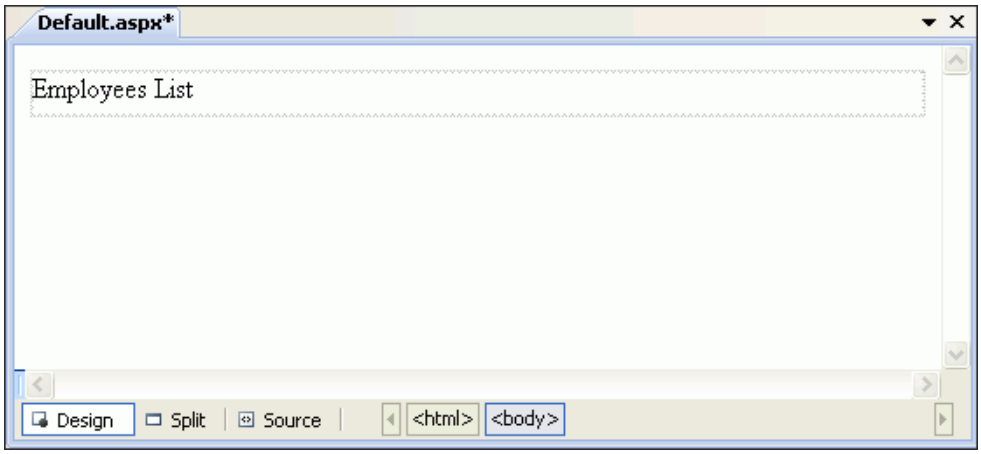
3. 「New Web Site」から、**「ASP.NET Web Site」**を選択し、Web サイトのディレクトリの場所を入力するか、または参照します。**「OK」**をクリックします。



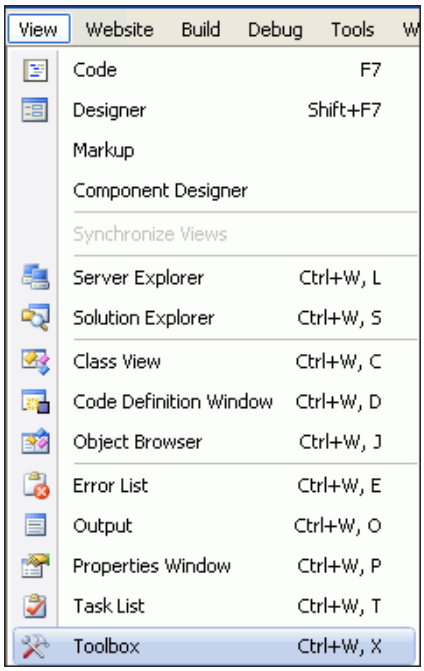
4. 「Default.aspx」タブで、画面の下部の「**Design**」アイコンをクリックします。



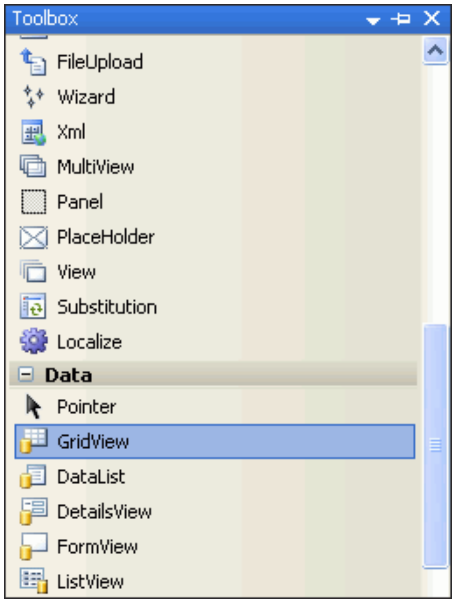
5. 点線の矩形として表示される 要素に、Employees List などのタイトルを入力します。



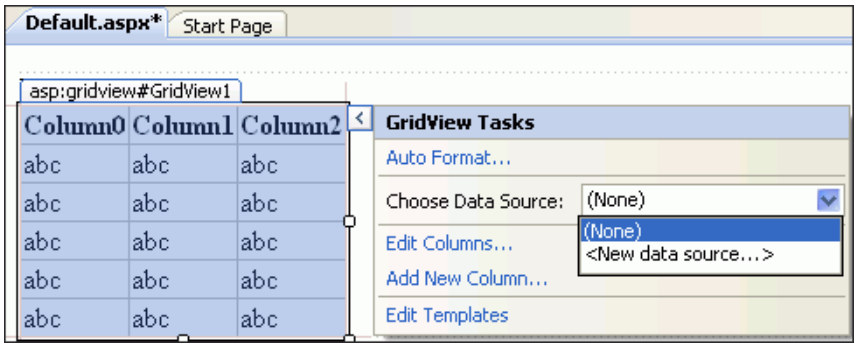
6. 「**View**」メニューから「**Toolbox**」を選択します。



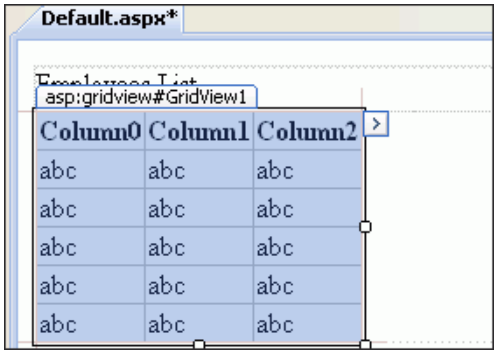
7. 「Data」グループを展開して、GridView コントロールをデザイナの `<div>` とラベル付けされた点線の矩形にドラッグします。



8. 仮のタイトルおよびコンテンツが含まれているグリッドが表示されると、その右側に GridView タスク・リストが表示されます。



このタスク・リストが表示されない場合は、グリッドを選択した後、右側の > 記号をクリックします。



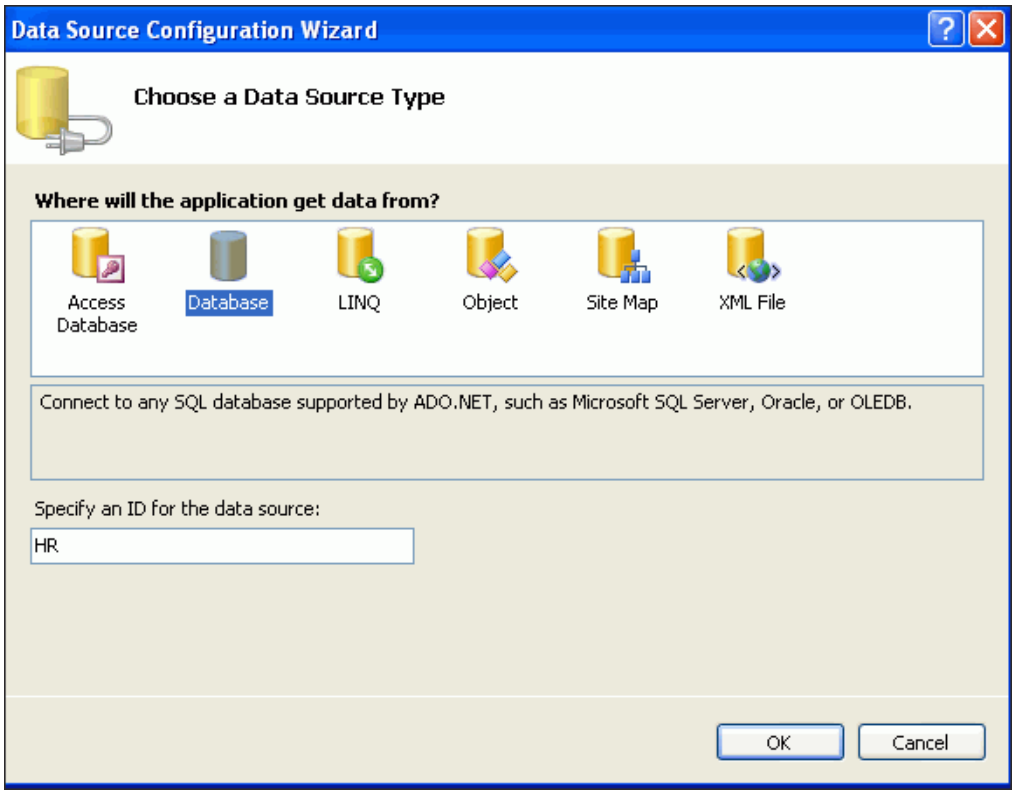
データソースの作成

データソースを作成するには、次の手順を実行します。

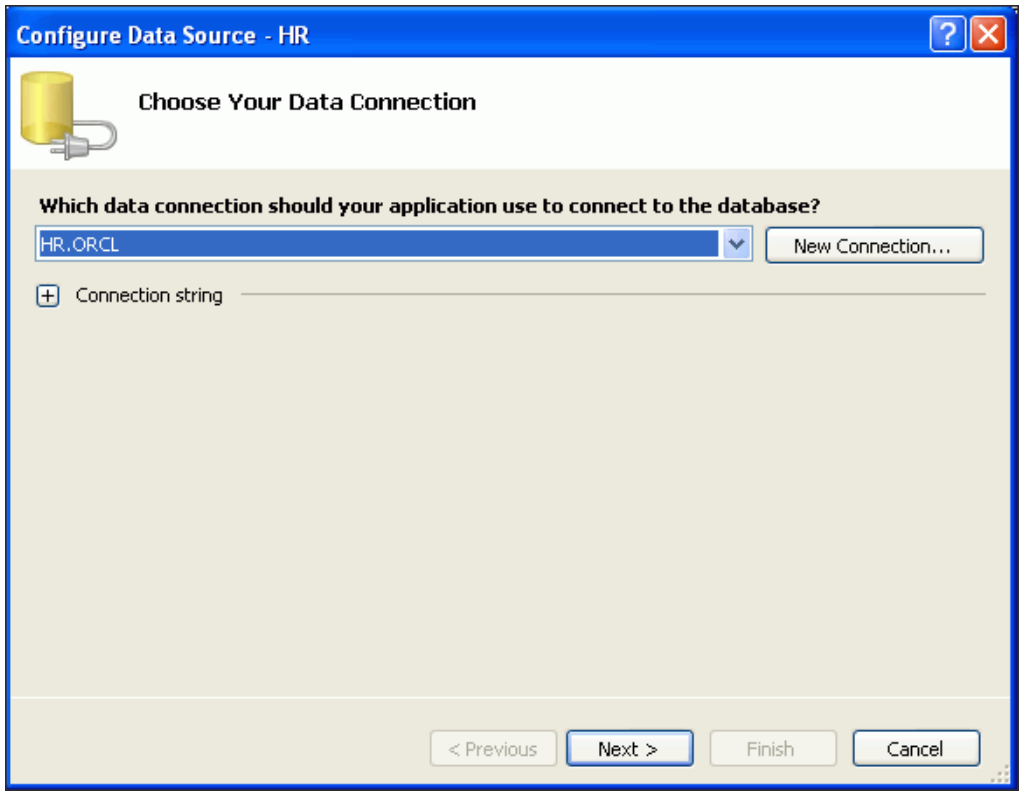
1. 「GridView Tasks」で、前の項の手順 8 に示す「Choose Data Source」リストから **<New data source...>** を選択します。

   「Data Source Configuration Wizard」が表示されます。

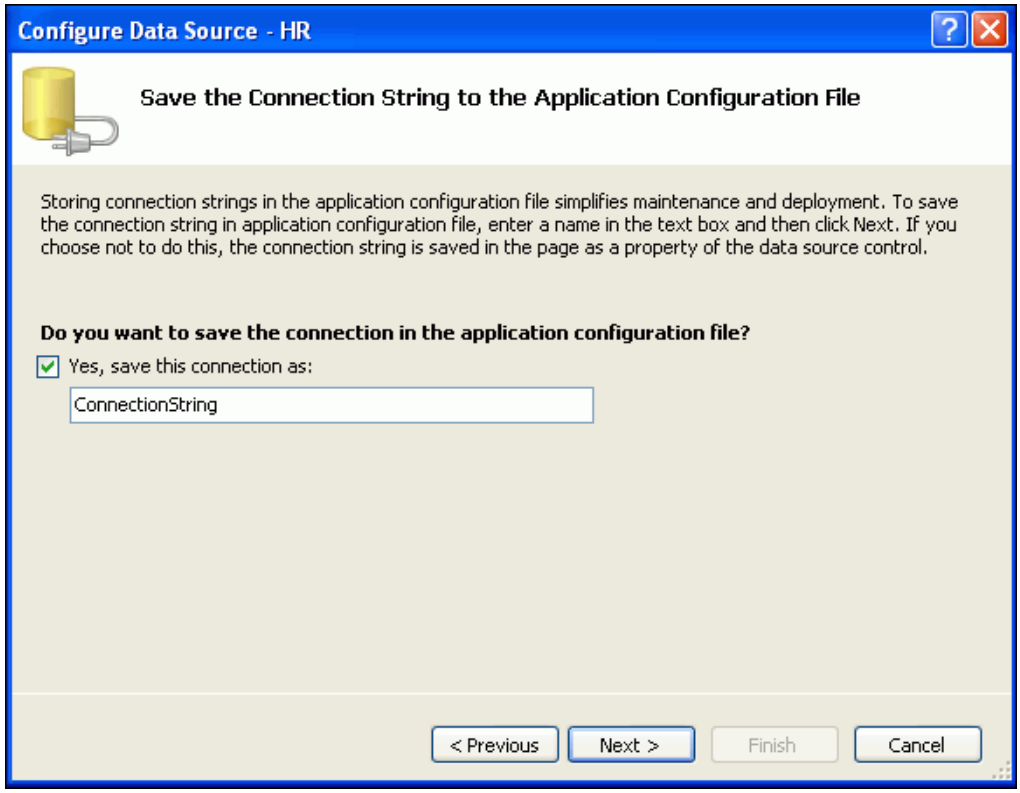
2. 「**Database**」を選択します。データソースの ID として HR と入力します。「**OK**」をクリックします。



3. 下矢印をクリックして、リストから HR.ORCL を選択します。「**Next**」をクリックします。



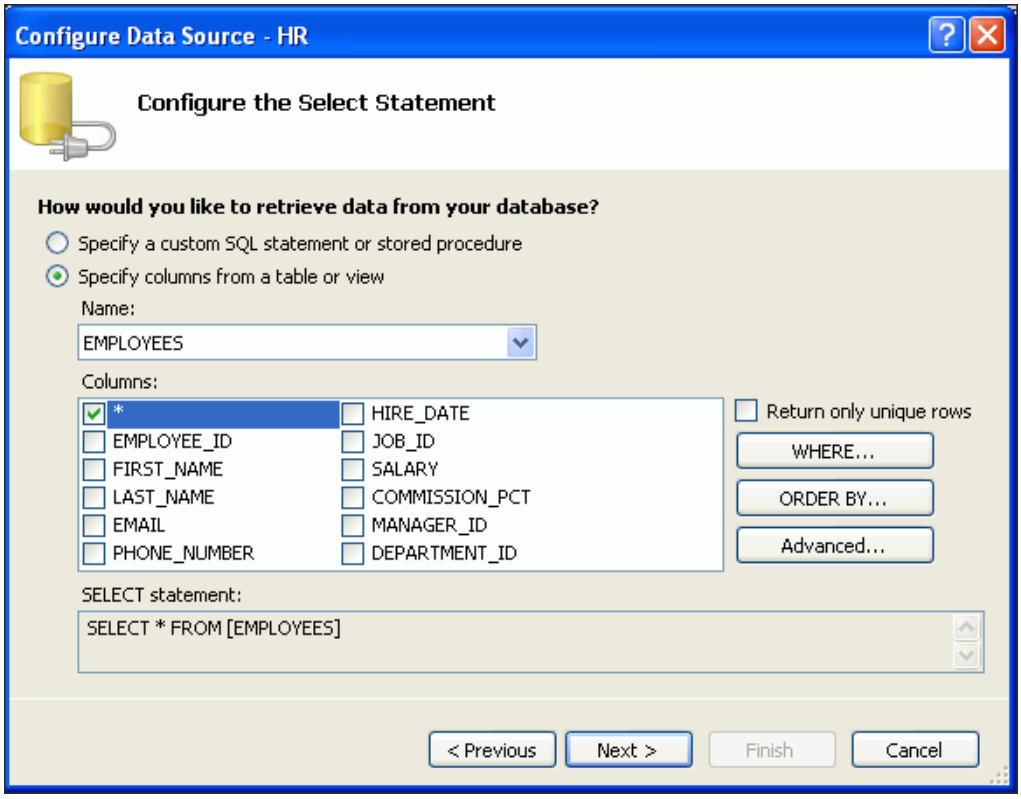
4. 「**Next**」をクリックして、接続文字列をアプリケーション構成ファイルに保存します。



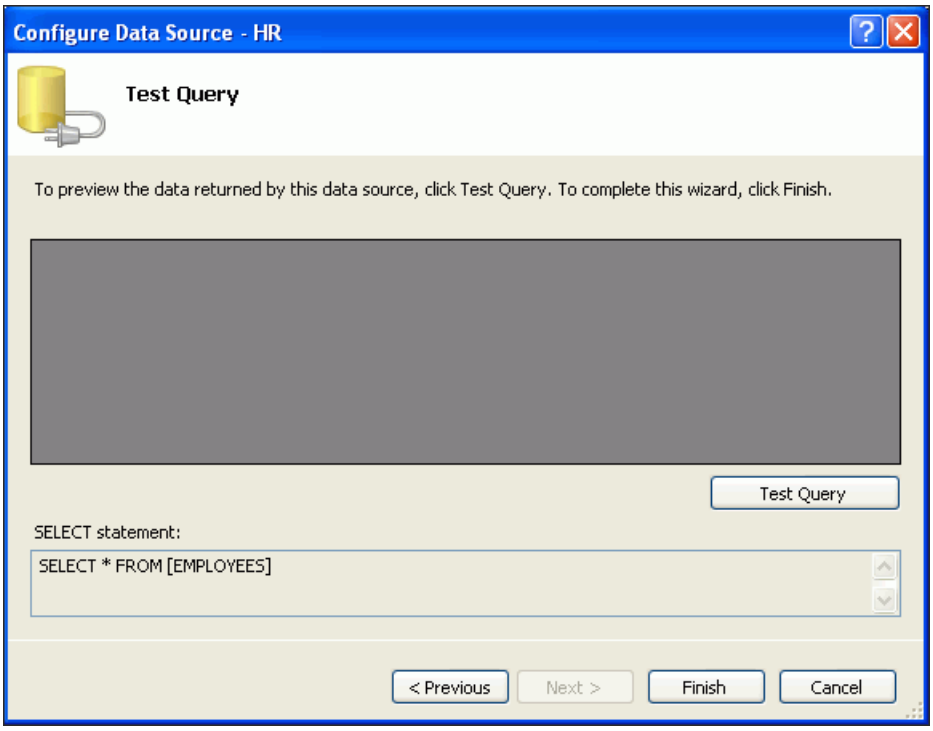
5. 「Name」リストから EMPLOYEES 表を選択します。「Columns」リストで、アスタリスク（*）の横のボックスを選択します。

   これらを選択すると、SELECT * FROM EMPLOYEES と入力した場合と同様に、Oracle によって EMPLOYEES 表からすべての行が戻されます。

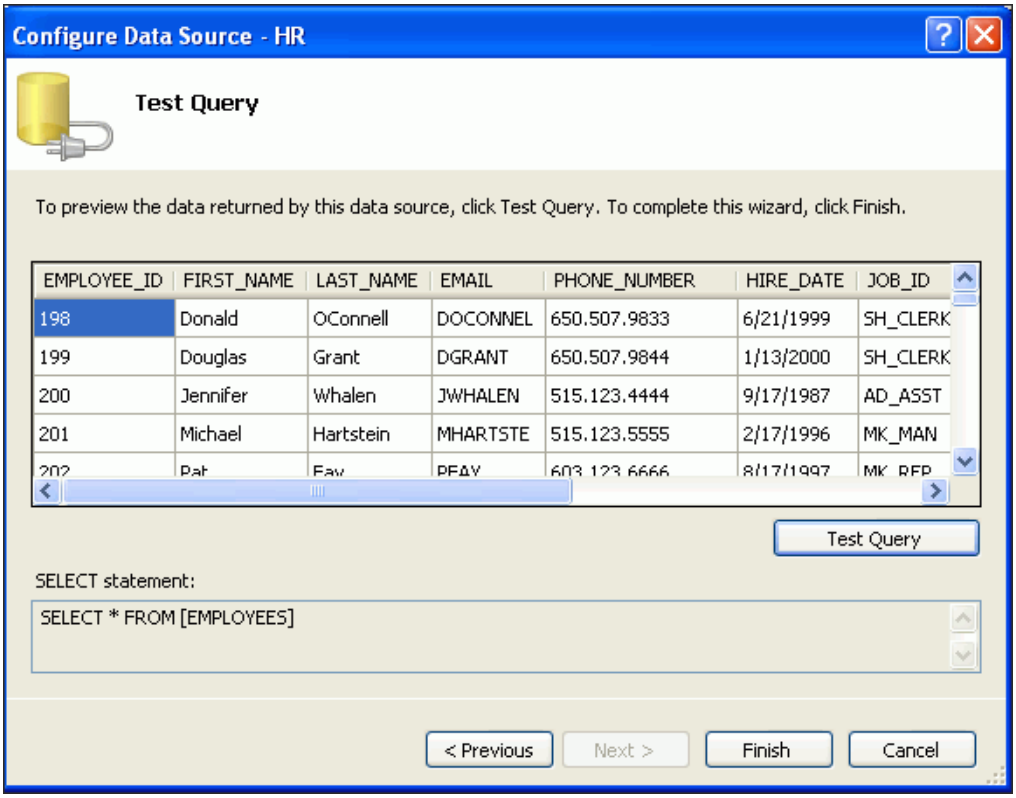
   「**Next**」をクリックします。



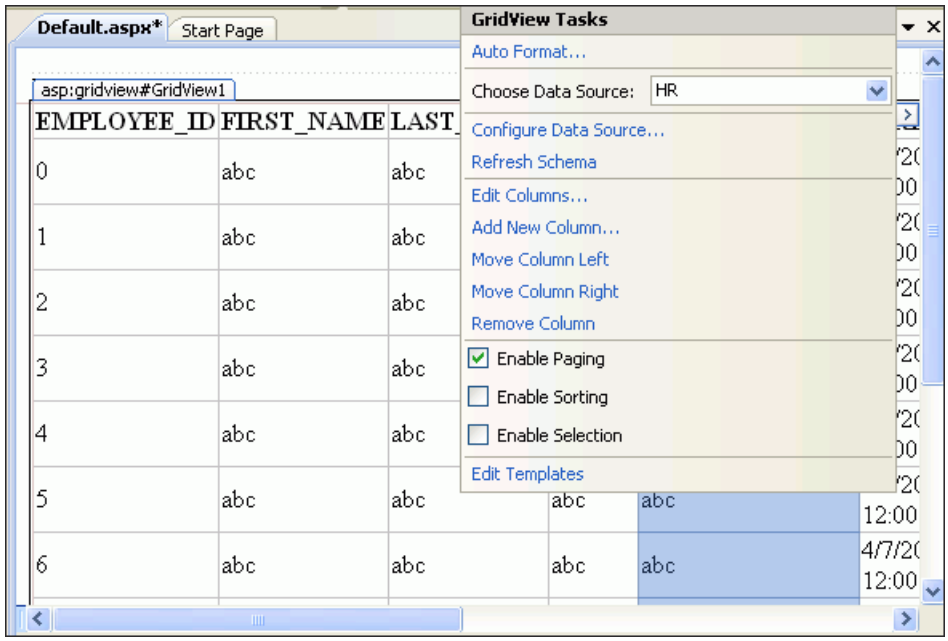
6. 「**Test Query**」をクリックします。



7. 「Test Query」の結果が表示されたら、「**Finish**」をクリックします。



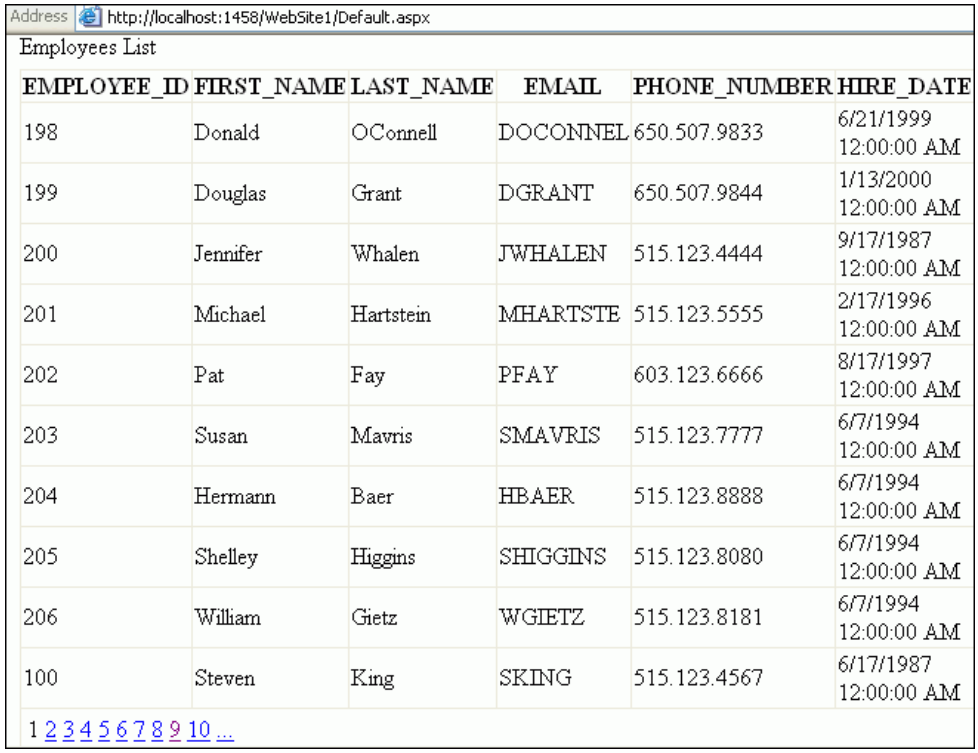
8. 「**GridView Tasks**」から、「Enable Paging」を選択します。このタスク・リストが表示されない場合は、グリッドを選択した後、右側の > 記号をクリックします。右にスクロールする必要がある場合があります。



9. 「**View**」メニューから、「**Solution Explorer**」を選択し、Web サイトを選択します。右クリックして「**Build Web Site**」を選択します。ステータス・バーに、成功または失敗が示されます。

10. 「View」メニューから、「Debug」→「Start Without Debugging」を選択します。

    次に示すようなブラウザ・ウィンドウが表示され、問合せでリクエストされたデータが示されます。ページの左下のセクションの番号を使用して、結果をページごとに確認できます。


Address http://localhost:1458/WebSite1/Default.aspx

Employees List

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE
198	Donald	OConnell	DOCONNEL	650.507.9833	6/21/1999 12:00:00 AM
199	Douglas	Grant	DGRANT	650.507.9844	1/13/2000 12:00:00 AM
200	Jennifer	Whalen	JWHALEN	515.123.4444	9/17/1987 12:00:00 AM
201	Michael	Hartstein	MHARTSTE	515.123.5555	2/17/1996 12:00:00 AM
202	Pat	Fay	PFAY	603.123.6666	8/17/1997 12:00:00 AM
203	Susan	Mavris	SMAVRIS	515.123.7777	6/7/1994 12:00:00 AM
204	Hermann	Baer	HBAER	515.123.8888	6/7/1994 12:00:00 AM
205	Shelley	Higgins	SHIGGINS	515.123.8080	6/7/1994 12:00:00 AM
206	William	Gietz	WGIETZ	515.123.8181	6/7/1994 12:00:00 AM
100	Steven	King	SKING	515.123.4567	6/17/1987 12:00:00 AM

1 2 3 4 5 6 7 8 9 10 ...

11. ブラウザを閉じます。

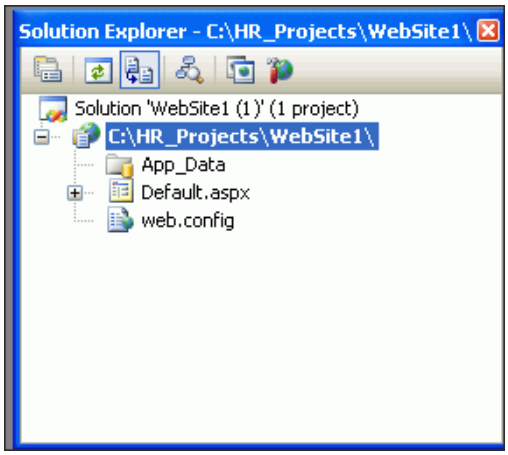
認証用の Web サイトの有効化

この項では、Web サイト認証を追加して従業員データにアクセスできるユーザーを制限する方法について説明します。ASP.NET ログイン・コントロールを使用し、Oracle Providers for ASP.NET で作成および格納したユーザーに対する検証によって認証を行います。

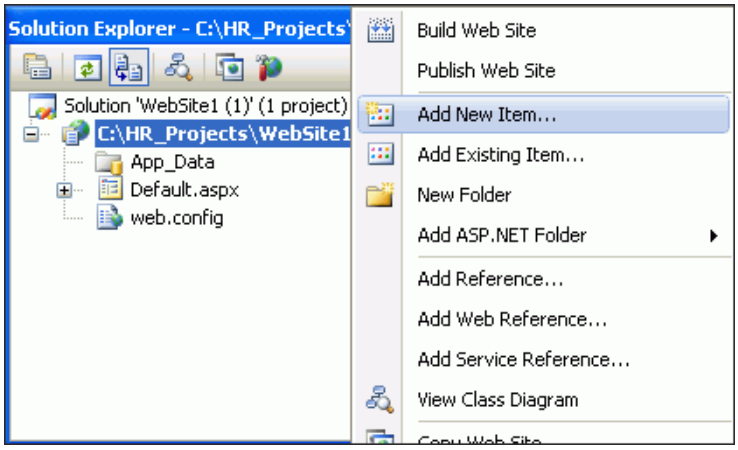
7-15 ページの「Oracle Providers for ASP.NET の有効化および軽量 Web ユーザーの作成」では、作成したばかりの ASP.NET アプリケーションを使用して、認可されたユーザーが情報にアクセスできるようにして従業員データを保護します。Oracle Providers for ASP.NET を使用してアプリケーションの Web ユーザーを作成します。この Web ユーザーは、ログイン・コントロールを使用して Web アプリケーションを起動し、証明が正しい場合に、従業員情報にアクセスできます。

1. 前の項で作成した Web サイトを再度開きます。

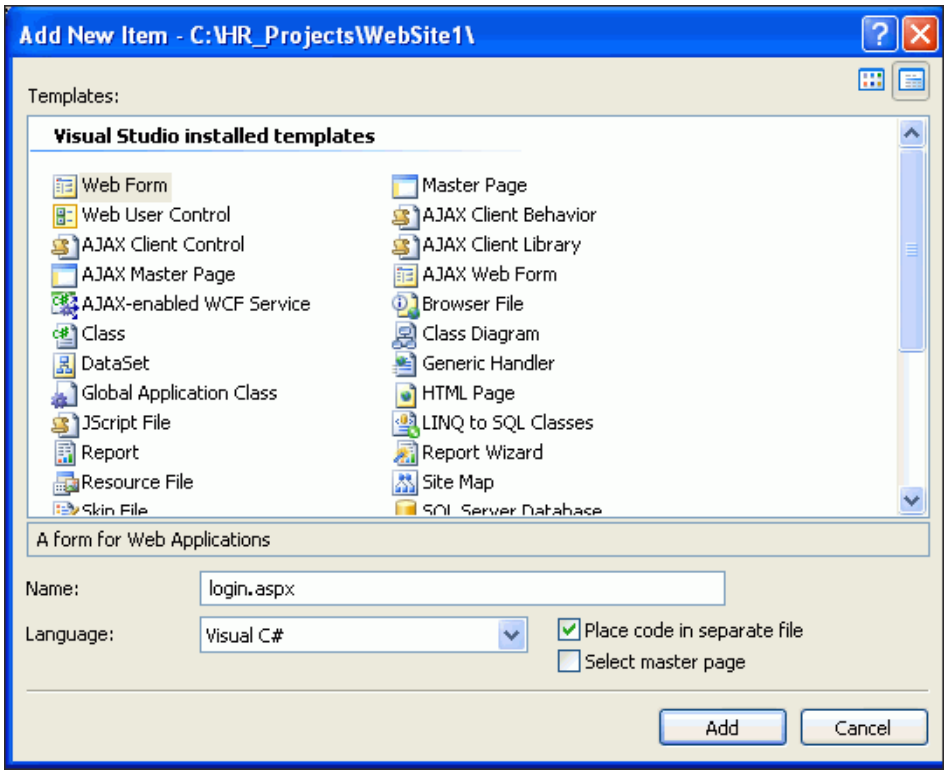
2. 「**View**」→「**Solution Explorer**」を選択して、Web サイトをクリックします。



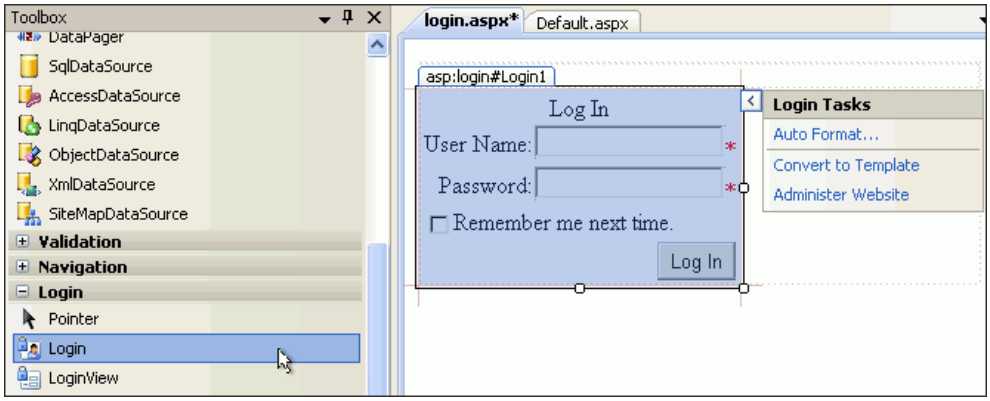
3. Web サイトを右クリックし、「**Add New Item**」を選択します。



4. 「Web Form」を選択して、名前 `login.aspx` を入力して、「**Add**」をクリックします。



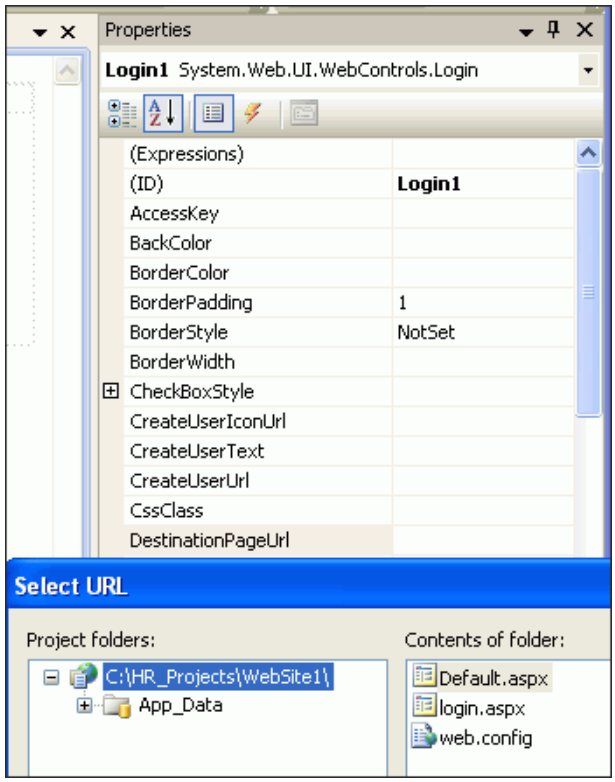
5. `login.aspx` ページが表示されたら、「Design」タブに切り替えます。
6. 「**View**」メニューから、「Toolbox」を開き、「Login」セクションを展開してログイン・コントロールをフォーム上の `<div>` とラベル付けされた点線の矩形にドラッグ・アンド・ドロップします。



これが標準 ASP.NET ログイン・コントロールです。これによって、Oracle Database に格納されているユーザー・ログイン資格証明を取得して検証できます。

7. ログイン・コントロールを右クリックして「Properties」を選択します。「DestinationPageUrl」で、Default.aspx を選択または入力します。

   ユーザーがログインに成功すると、従業員データが含まれる Default.aspx ページが表示されます。ログインに成功しなかった場合は、ログイン・ページにリダイレクトされます。



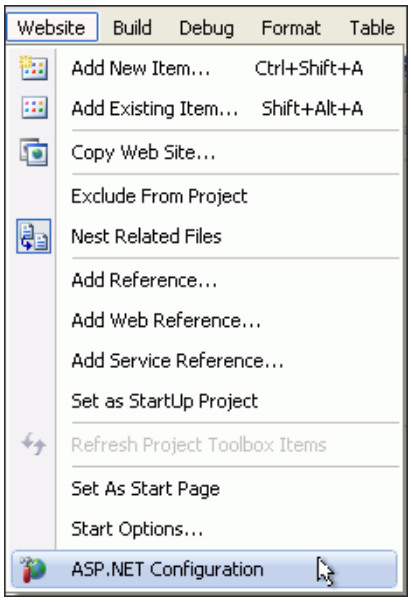
Oracle Providers for ASP.NET の有効化および軽量 Web ユーザーの作成

この項では、ASP.NET Web サイト管理ツールを使用して次の操作を行います。

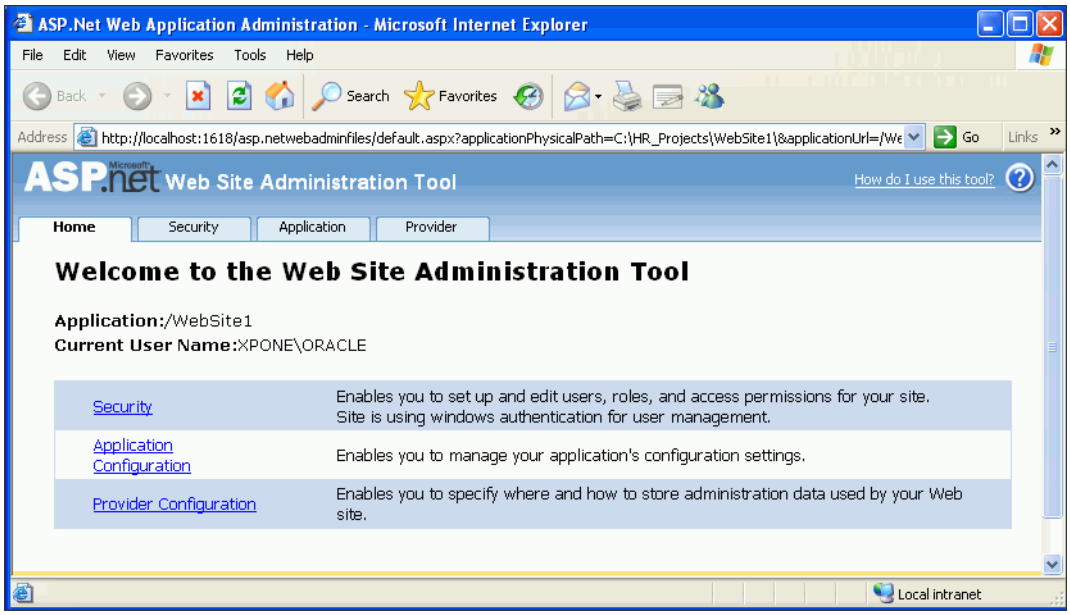
- Web サイトで Oracle ASP.NET プロバイダを使用するための指示
- この Web サイトに固有の新しい Web ユーザーの作成および 7-12 ページの「認証用の Web サイトの有効化」でサイトに追加された認証機能のデモ

Web サイトで Oracle ASP.NET プロバイダを使用して新しい Web ユーザーを作成するには、次の手順を実行します。

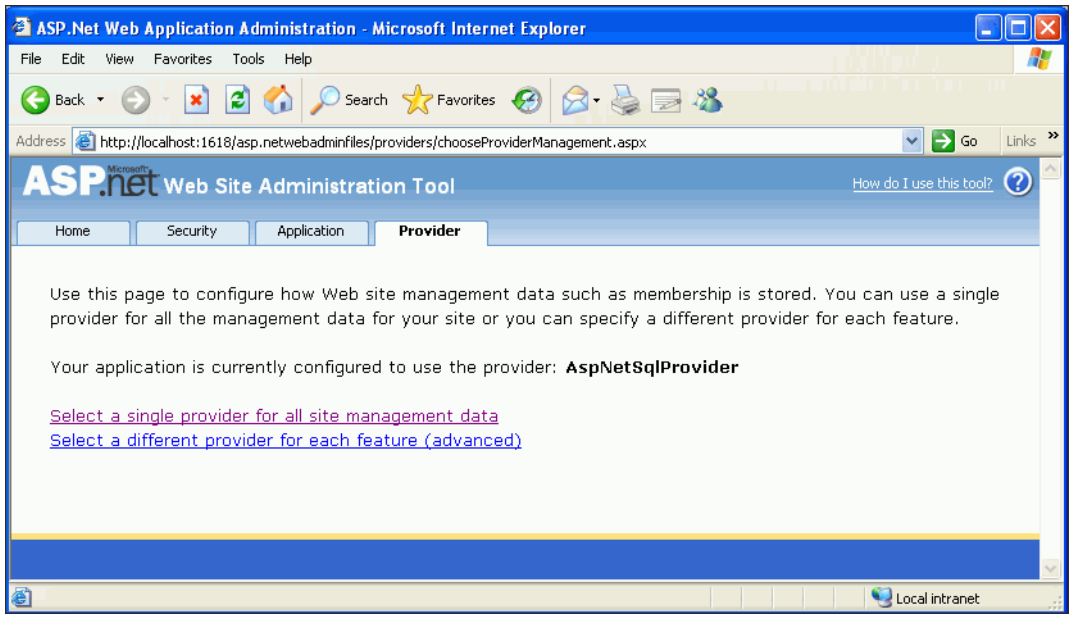
1. Visual Studio で、「**Website**」→「**ASP.NET Configuration**」を選択します。



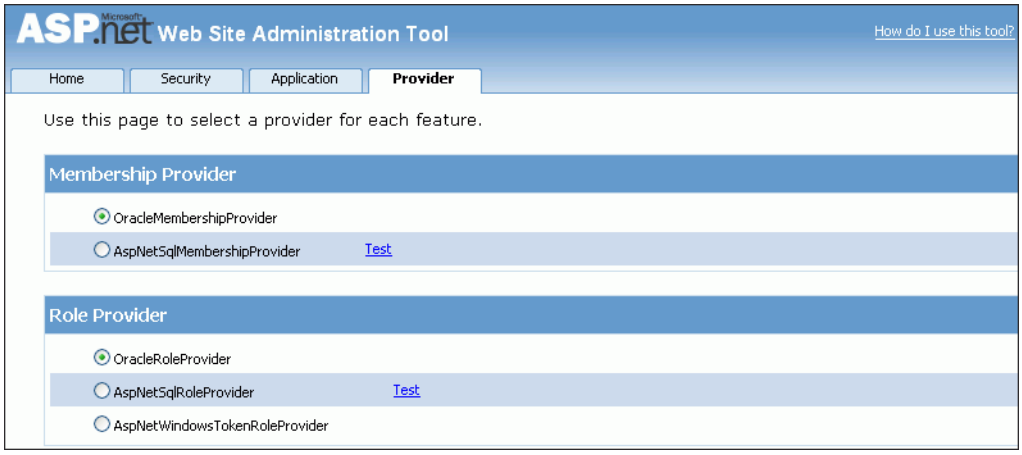
2. ASP.NET Web サイト管理ツールが表示されたら、「Provider」タブを選択します。



3. 「Provider」ページで、2 番目のリンク「Select a different provider for each feature (advanced)」を選択します。



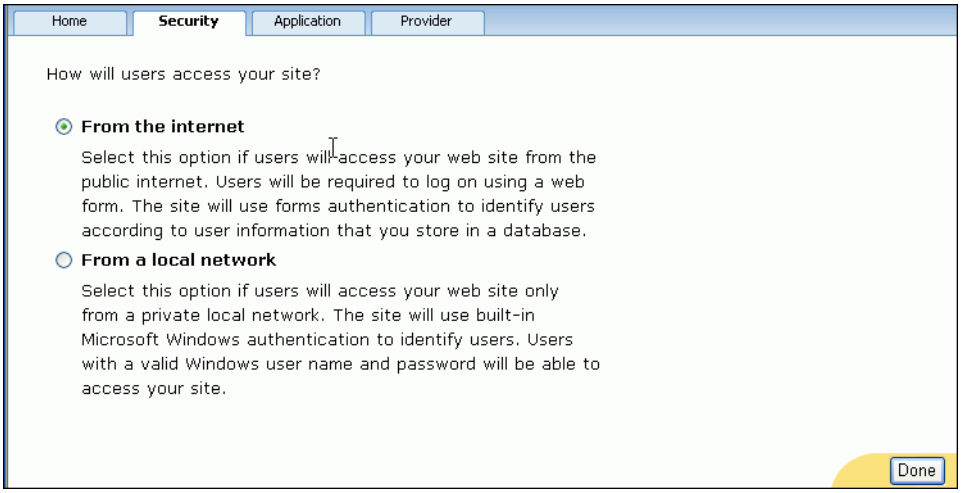
4. 「Provider」ページが再度表示されたら、メンバーシップ・プロバイダおよびロール・プロバイダを Oracle バージョンに変更します（これらのプロバイダが選択されていない場合）。



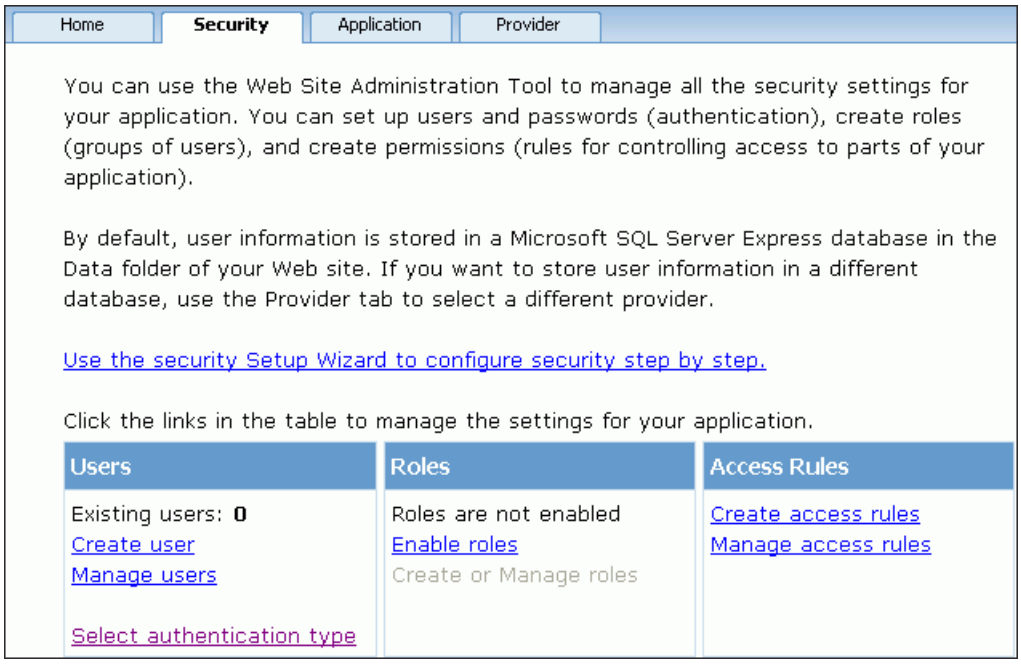
5. 「Security」タブに移動し、「Users」で**「Select authentication type」**をクリックします。

   デフォルトでは、ASP.NET サイトでユーザーの識別に Windows 認証が使用されます。サイト固有のログインおよびパスワードでユーザーを識別する Web サイトを作成します。したがって、ログインおよびパスワードの使用に対応するようにサイトを構成する必要があります。

6. 「Security」ページが再度表示されたら、**「From the internet」**を選択して**「Done」**をクリックします。

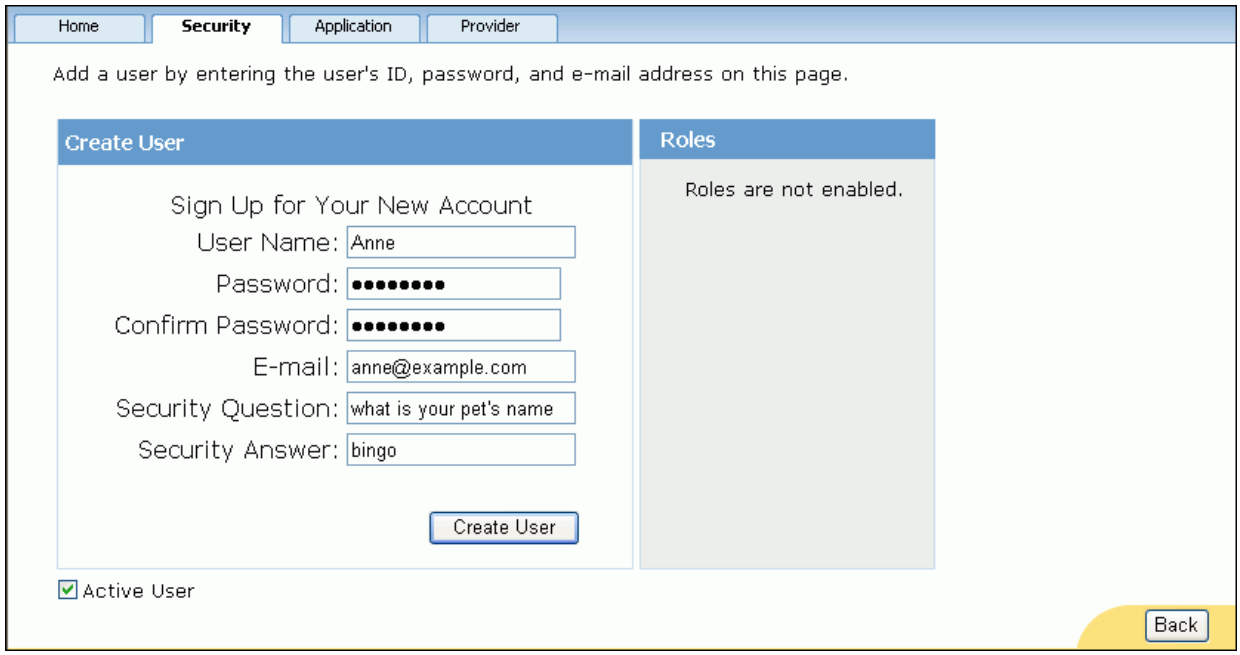


7. 「Users」に新しいリンクが示された「Security」タブが再度表示されたら、**「Create user」**を選択します。

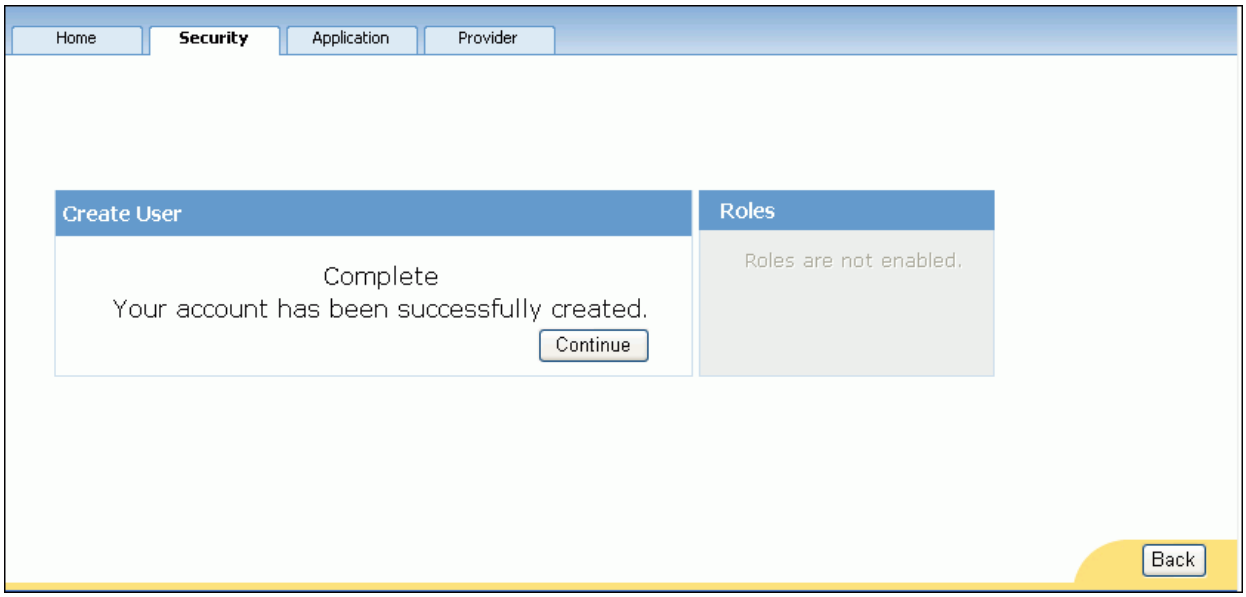


8. 「Create User」セクションで、Web サイトへのアクセスを許可するユーザーの情報を次のように入力します。パスワードは、英数字以外の文字を 1 文字含む 7 文字以上で指定します。

   「Create User」をクリックします。

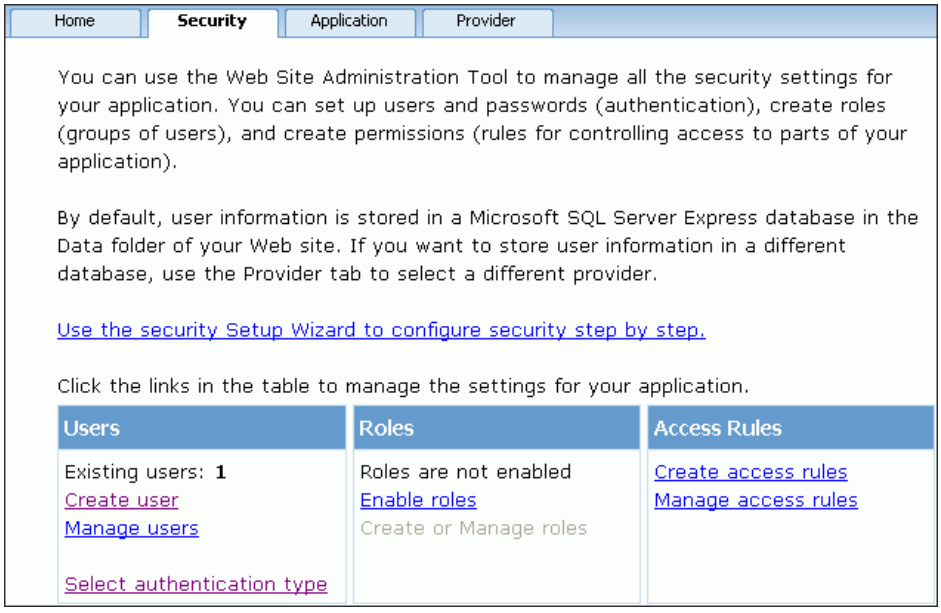


9. アカウントが正常に作成されたことを示す「Security」ページが再度表示されたら、**「Security」**タブをクリックします。



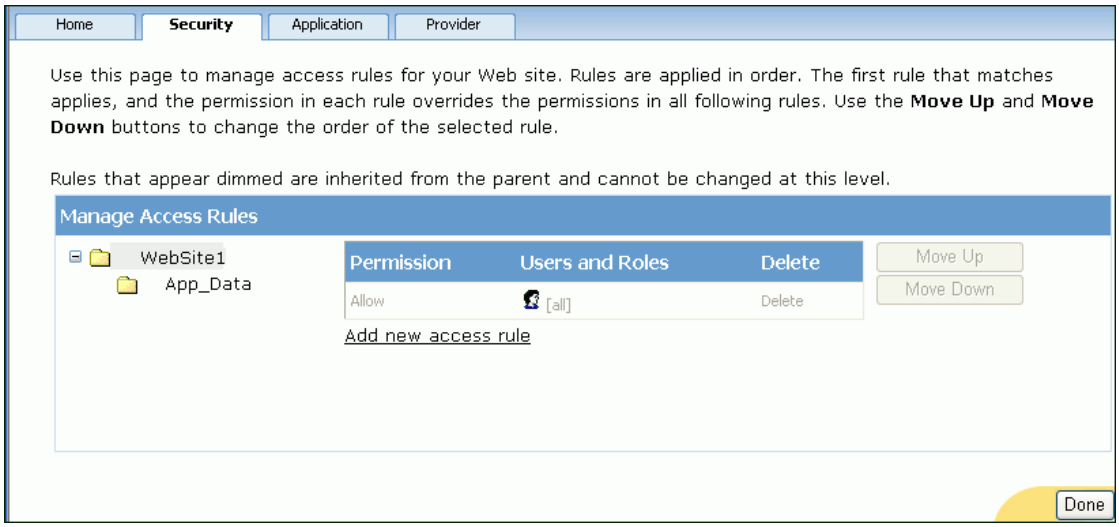
注意：この画面には、他のユーザーの作成を続けるオプションや、別のタブに移動するオプションがあります。

10. 「Security」メイン・ページが再度表示されたら、「Access Rules」で、**「Manage access rules」**を選択します。



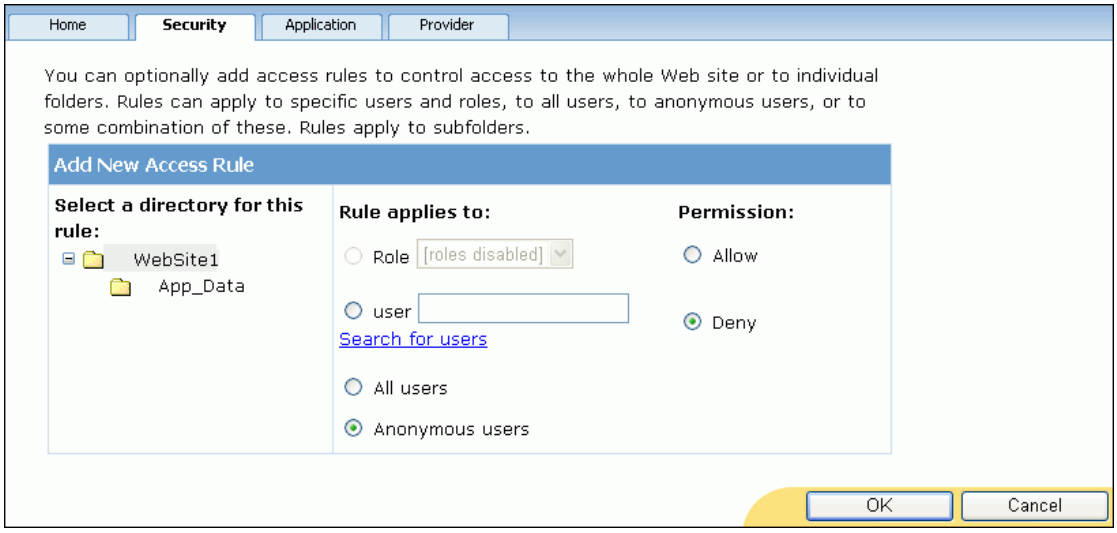
注意：現在、「Users」には既存ユーザーが 1 人表示されています。

11. 「**Security**」タブに「Manage Access Rules」セクションが表示されたら、「**Add new access rule**」をクリックします。



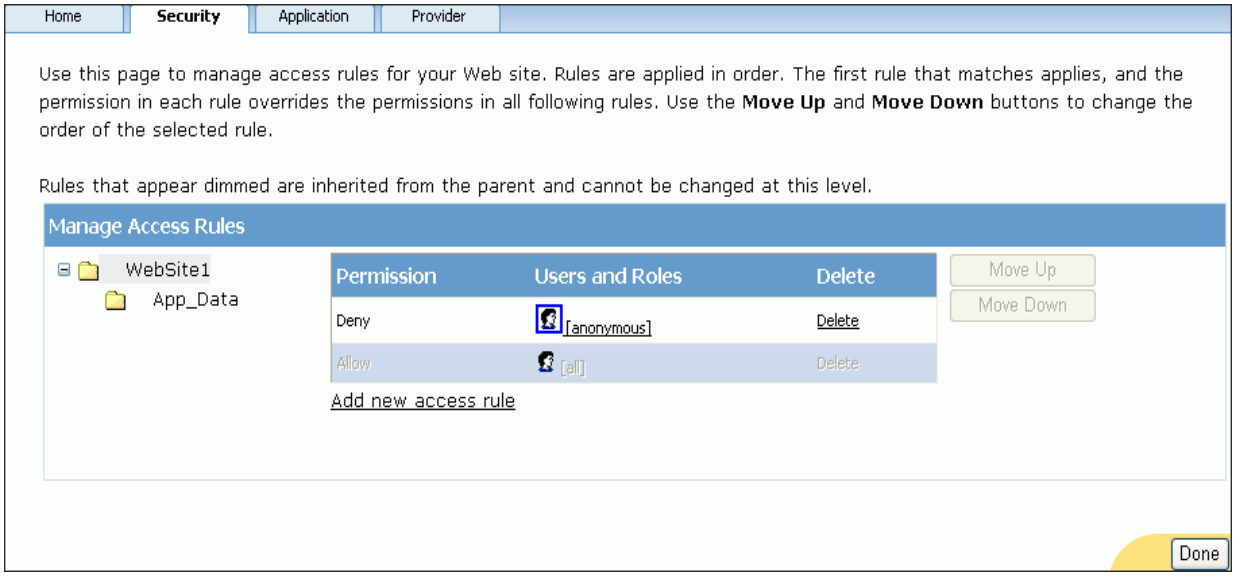
12. 「**Anonymous users**」および「**Deny**」を選択してから「**OK**」をクリックします。

デフォルトでは、Web サイトへの匿名アクセスが有効になっています。前述の設定で、匿名アクセスを無効にすることによって Web サイトを保護します。これで、認証されたユーザーのみが従業員データを表示できるようになります。



13. 現在「Security」ページは、サイトへの匿名ユーザーのアクセスが Web サイトによって拒否されることを示しています。

    「**Done**」をクリックします。



14. ブラウザを閉じます。

Web サイト認証のテスト

この Web サイトに固有の Web ユーザーが作成されたので、Web サイトでは、このユーザーによる従業員データへのアクセスが許可され、匿名ユーザーを含むその他のすべてのユーザーによるアクセスは拒否されるようになります。

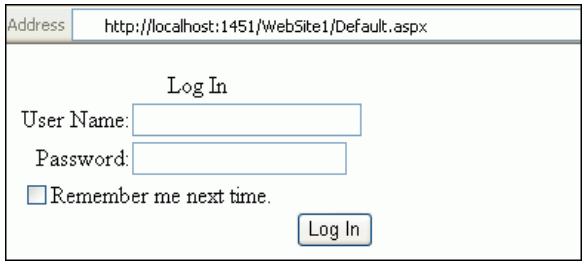
この項では、匿名ユーザー、不正ユーザー、間違ったパスワードを使用する認可されたユーザー、正しいパスワードを使用する認可されたユーザーとして従業員データにアクセスしてみます。最後のシナリオでのみ、従業員データへのアクセス権が Web サイトから付与されます。

注意： ASP.NET プロバイダ・ユーザーに対して 10 分間に 5 回以上連続して無効なパスワードを入力すると、不正ユーザーがパスワードを推測してアクセスできないようにアカウントがロックされます。Oracle Membership Provider では、`MaxInvalidPasswordAttempts`（デフォルト：5 回）および `PasswordAttemptWindow`（デフォルト：10 分）というプロパティによってこれらのセキュリティ・メジャーを設定します。これらのプロパティは、`machine.config` ファイルまたは `web.config` ファイルで変更できます。

アカウントがロックされた場合は、`UnlockUser` メソッドをコールしてユーザーのロックを解除できます。

---

1. 「**Debug**」メニューから、「**Start Without Debugging**」を選択し、ログイン Web ページが表示されたら、末尾が `login.aspx` ではなく `Default.aspx` になるように URL を変更して [Enter] キーを押します。



アクセスが拒否され、ログイン・ページにリダイレクトされます。これは、匿名ユーザーが Web サイトを参照できず、資格証明を持つユーザーのみがアクセスできることを示します。

認証メカニズムを試す場合は、この手順を繰り返すか、または様々な操作を試してみます。様々な操作で、新しいブラウザを起動するか、またはブラウザ・キャッシュを消去します。Web ページはブラウザにキャッシュされるため、`Default.aspx` に再度アクセスすると、キャッシュされたその Web ページが表示される場合があります。これは意図された動作ではなく、Web ページは、新しいブラウザ・インスタンスを使用するか、またはブラウザ・キャッシュを消去することによって完了する ASP.NET プロバイダ認証プロセスを実行する必要があります。

2. URL の `login.aspx` の後のテキストを削除します。これによって、URL は最初にサイトにアクセスしたときの元の状態に戻ります。

   ユーザー名 Bob、および英数字以外の文字を 1 文字含む 7 文字以上のパスワードを入力します。

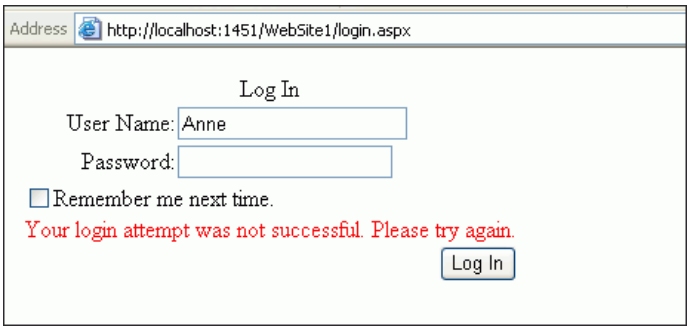
   「**Log In**」をクリックします。

「Your login attempt was not successful. Please try again.」というメッセージが示されたページが表示されます。

Bob は認可されたユーザーではありません。このユーザーによるアクセスは Web サイトによって適切に拒否されました。

3. ユーザー名 Anne を、この Web サイト・ユーザーに対しては無効なパスワードを使用して入力します。

   「Log In」をクリックします。



   スクリーンショットに示すように、このユーザーによるアクセスは拒否され、このユーザーの資格証明が Oracle Membership Provider に格納された資格証明では検証できなかったことが示されます。

4. この Web サイト・ユーザーの正しいパスワードを入力します。

   「Log In」をクリックします。

   従業員データが表示されます。これによって、認可されたユーザーのみがデータにアクセスできることが示されます。このように、Oracle Providers for ASP.NET では、非常に単純な方法で Web サイトのセキュリティが提供されています。

これで、データ駆動型の ASP.NET Web アプリケーションが作成されました。このアプリケーションによって、認証が行われ、データベースから従業員データが取得されます。

8

.NET ストアド・プロシージャの開発とデプロイ

この章の内容は次のとおりです。

.NET ストアド・プロシージャの概要

.NET ストアド・プロシージャは、SQL 文または PL/SQL 文を含む .NET 言語で記述されたメソッドまたはプロシージャです。

カスタム・ストアド・プロシージャおよびファンクションは、C# や VB.NET などの .NET に準拠した任意の言語を使用して記述できます。また、これらの .NET ストアド・プロシージャは、他の PL/SQL ストアド・プロシージャまたは Java ストアド・プロシージャと同様にデータベースで使用できます。.NET ストアド・プロシージャは、PL/SQL パッケージ、プロシージャ、ファンクションおよびトリガーからコールできます。また、SQL 文からコールすることも、PL/SQL プロシージャまたはファンクションをコールできる任意の場所からコールすることもできます。

この章の例を実行するには、Oracle Database Extensions for .NET（.NET ストアド・プロシージャを記述できるデータベース・オプション）をデータベースにインストールして構成しておく必要があります。

この章では、.NET ストアド・プロシージャをアプリケーションで使用およびデプロイする方法を説明します。

共通言語ランタイム・サービスの開始

.NET ストアド・プロシージャを使用するには、最初に共通言語ランタイム・エージェント（`OraClrAgent` サービス）を開始する必要があります。このサービスはデフォルトでは開始されません。このサービスは、クライアント上ではなく Oracle Database 上にあることに注意してください。

注意： `OraClrAgnt` は、`Oracle`*`OracleHomeName`*`ClrAgnt` という名前で「コントロール パネル」の「サービス」からアクセスできます。*`OracleHomeName`* は Oracle ホームです。

共通言語ランタイム・サービスを開始するには、次の手順を実行します。

1. **「スタート」**メニューから**「すべてのプログラム」**、**「管理ツール」**、**「サービス」**の順に選択します。
2. 「サービス」ウィンドウで**「拡張」**タブをクリックします。

   サービスのリストをスクロールし、「`Oracle`*`OracleHomeName`*`ClrAgnt`」を選択します。
3. **「サービスの開始」**ハイパーリンクをクリックします。

   「サービス コントロール」ウィンドウに、`OracleClrAgent` を開始していることが表示されます。
4. 「サービス コントロール」ウィンドウが閉じたら、`OracleClrAgent` の状態が「開始」に変わっていることを確認します。

SYSDBA としての接続の作成

次に、SYSDBA としてデータベース接続を作成する必要があります。これにより、Oracle プロジェクトをデプロイできるようになります。

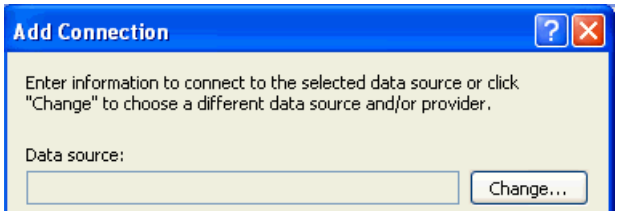
注意： このタスクを実行するには、SYSDBA としての管理者権限が必要です。

注意： Enterprise Manager を使用して sys アカウントのパスワードを設定する場合は、『Oracle Database 2 日でデータベース管理者』の管理者アカウントおよび管理者権限に関する項を参照してください。

ODT でデータベース接続を作成するには、次の手順を実行します。

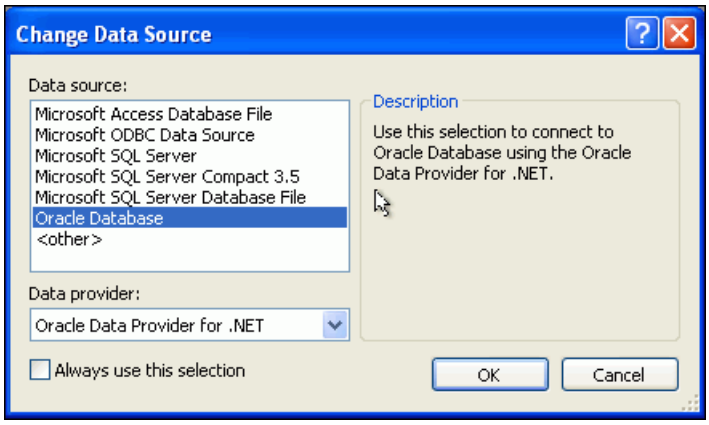
1. 「View」メニューから**「Server Explorer」**を選択します。
2. Server Explorer で、**「Data Connections」**を右クリックします。
3. **「Add Connection」**を選択します。
4. 「Add Connection」ウィンドウが表示されたら、「Data source」に「Oracle Database (Oracle ODP.NET)」と表示されているかどうかを確認します。

   表示されている場合は、手順 6 に進みます。



「Data source」に「Oracle Database (Oracle ODP.NET)」と表示されていない場合は、**「Change」**を選択します。

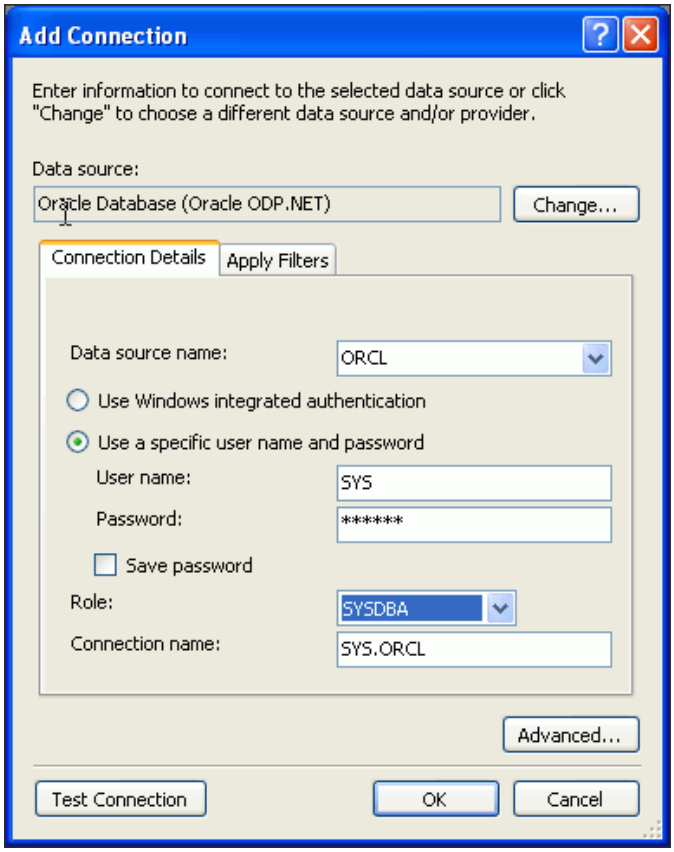
「Change Data Source」ウィンドウが表示されます。



5. 「Oracle Database」を選択してから、「Oracle Data Provider for .NET」を選択します。
6. 「Add Connection」ウィンドウでは、次のように指定します。
   - 「**User name**」には sys と入力します。
   - 「**Password**」には、sys アカウントのロック解除および設定を行った管理者により設定されたパスワードを入力します。

     Enterprise Manager を使用して sys アカウントのパスワードを設定する場合は、『Oracle Database 2 日でデータベース管理者』の管理者アカウントおよび管理者権限に関する項を参照してください。
   - 「**Role**」が Sysdba に設定されていることを確認します。

     「**Connection name**」は「**Data source name**」および「**User name**」の値から自動生成されます。



7. 「Add Connection」ウィンドウで、「**OK**」をクリックします。

   これで、「Server Explorer」ウィンドウに SYS.ORCL 接続が含まれます。

Oracle プロジェクトの作成

.NET でストアド・プロシージャを使用するには、ストアド・プロシージャを保持するための新しい Oracle プロジェクトを最初に作成する必要があります。

.NET ストアド・プロシージャ用のプロジェクトを作成するには、次の手順を実行します。

1. 「**File**」メニューから「**New**」、「**Project**」の順に選択します。

   「New Project」ダイアログ・ボックスが表示されます。

2. 「Project Types」で、作成するプロジェクトの種類を選択します。

   - **Visual C#:**

     「**Visual C#**」、「**Database**」の順に選択し、「Templates:」の下で「**Oracle Project**」を選択します。

     「**Name:**」に `HR_DeployStored_CS` と入力します。

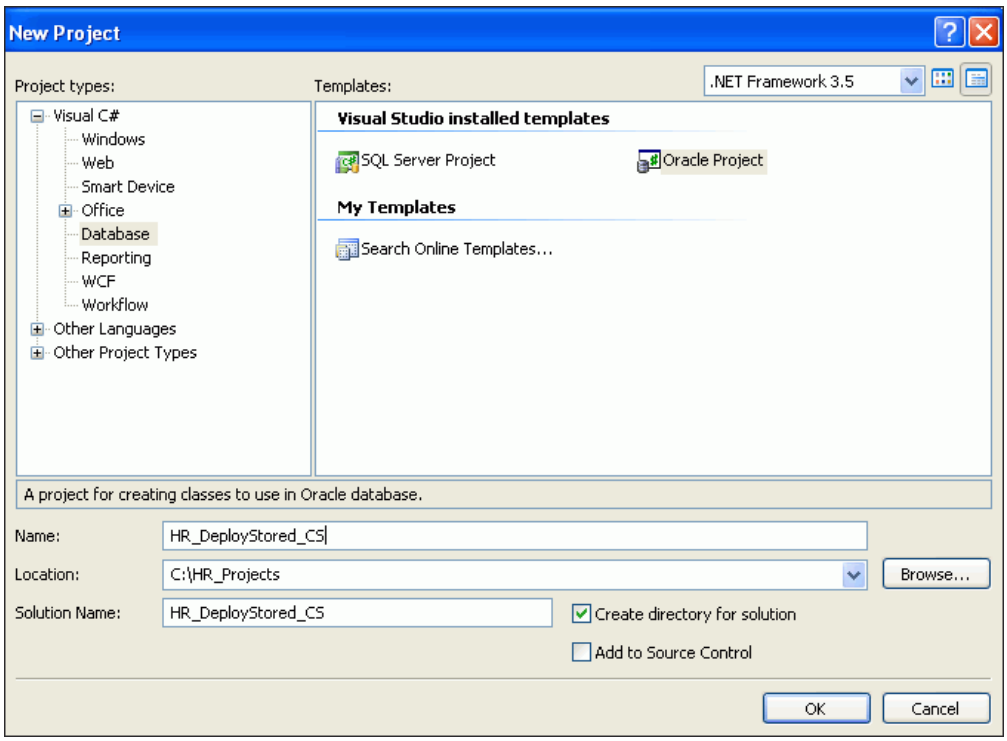
   - **Visual Basic:**

     「**Other Languages**」、「**Visual Basic**」、「**Database**」の順に選択し、「Templates:」の下で「**Oracle Project**」を選択します。

     「**Name:**」に `HR_DeployStored_VB` と入力します。

3. 「**Location:**」に `C:¥HR_Projects` と入力します。

4. 「**OK**」をクリックします。



.NET ストアド・ファンクションおよびプロシージャの作成

これで、.NET ストアド・プロシージャを作成する準備ができました。

.NET ストアド・プロシージャを作成するには、次の手順を実行します。

1. 「Solution View」で、プロジェクトの**「Class1.cs」**タブまたは**「Class1.vb」**タブを選択します。
2. 3-6 ページの「名前空間ディレクティブの追加」の説明に従って、言語ごとに次の名前空間ディレクティブを追加します。

   Visual C#:

   ```
   using Oracle.DataAccess.Client;
   using Oracle.DataAccess.Types;
   ```

   Visual Basic:

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

3. 3-4 ページの「参照の追加」の説明に従って、`Oracle.DataAccess.dll` に参照を追加します。
4. 次に示すように、`getDepartmentno()` メソッドを `Class1` 宣言にコピーします。

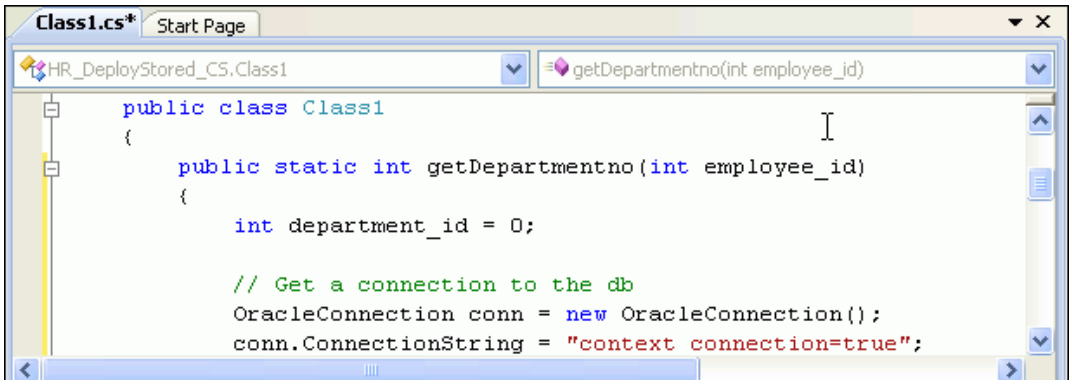
   Visual C#

   ```
   public static int getDepartmentno(int employee_id)
   {
     int department_id = 0;

     // Get a connection to the db
     OracleConnection conn = new OracleConnection();
     conn.ConnectionString = "context connection=true";
     conn.Open();

     // Create and execute a command
     OracleCommand cmd = conn.CreateCommand();
     cmd.CommandText = "select department_id from employees where employee_id = :1";
     cmd.Parameters.Add(":1", OracleDbType.Int32, employee_id,
       ParameterDirection.Input);
     OracleDataReader rdr = cmd.ExecuteReader();

     while(rdr.Read())
       department_id=rdr.GetInt32(0);

     rdr.Close();
     cmd.Dispose();

     // Return the employee's department number
     return department_id;
   }
   ```

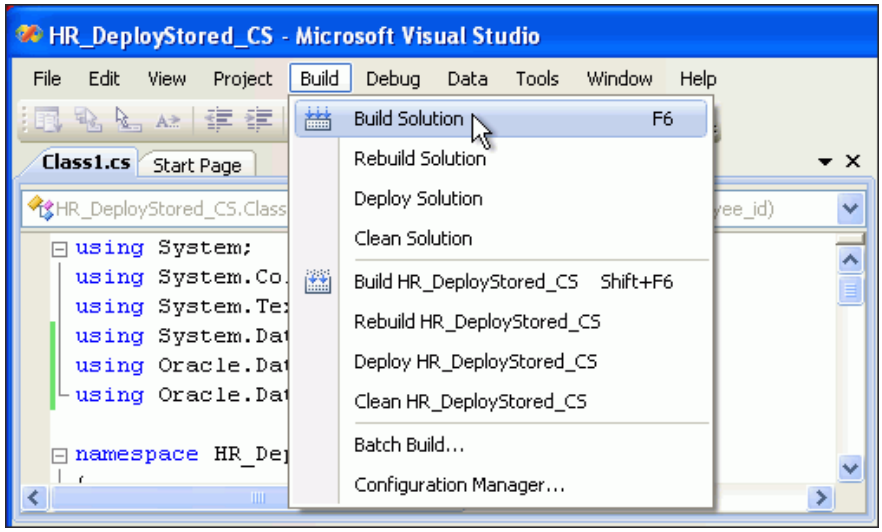
   Visual Basic:

   ```
   Public Shared Function getDepartmentno(ByVal employee_id As Integer) As Integer
     Dim department_id As Integer = 0

     ' Get a connection to the db
     Dim conn As OracleConnection = New OracleConnection()
     conn.ConnectionString = "context connection=true"
     conn.Open()

     ' Create and execute a command
   ```

```
  Dim cmd As OracleCommand = conn.CreateCommand()
  cmd.CommandText = "select department_id from employees where employee_id = :1"
  cmd.Parameters.Add(":1", OracleDbType.Int32, employee_id,
    ParameterDirection.Input)
  Dim rdr As OracleDataReader = cmd.ExecuteReader()

  While rdr.Read()
    department_id = rdr.GetInt32(0)

  End While

  rdr.Close()
  cmd.Dispose()

  ' Return the employee's department number
  Return department_id

End Function
```



5. Class1 を保存します。
6. 「**Build**」メニューから「**Build Solution**」を選択します。



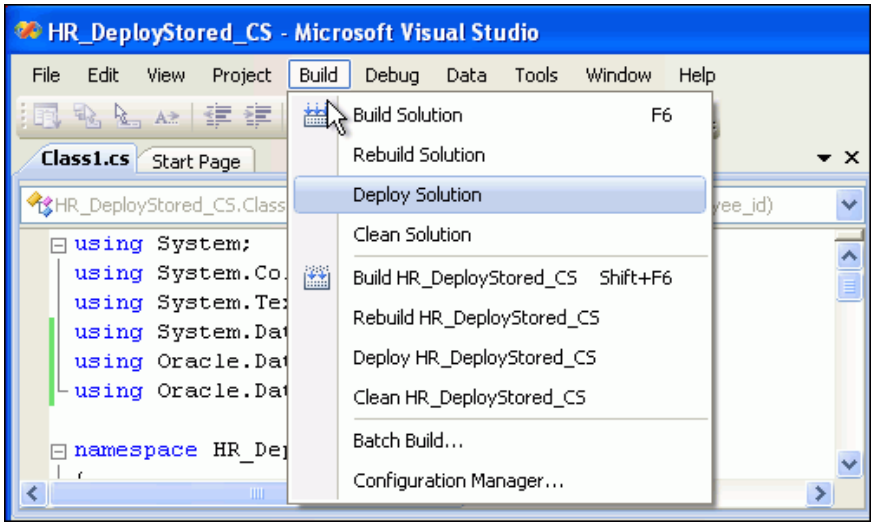
7. 正しくビルドされたことが「Output」ウィンドウに示されていることを確認し、このウィンドウを閉じます。

.NET ストアド・ファンクションおよびプロシージャのデプロイ

これまでの手順により、8-6 ページの「.NET ストアド・ファンクションおよびプロシージャの作成」で作成した .NET ストアド・プロシージャをデプロイできる状態になっています。

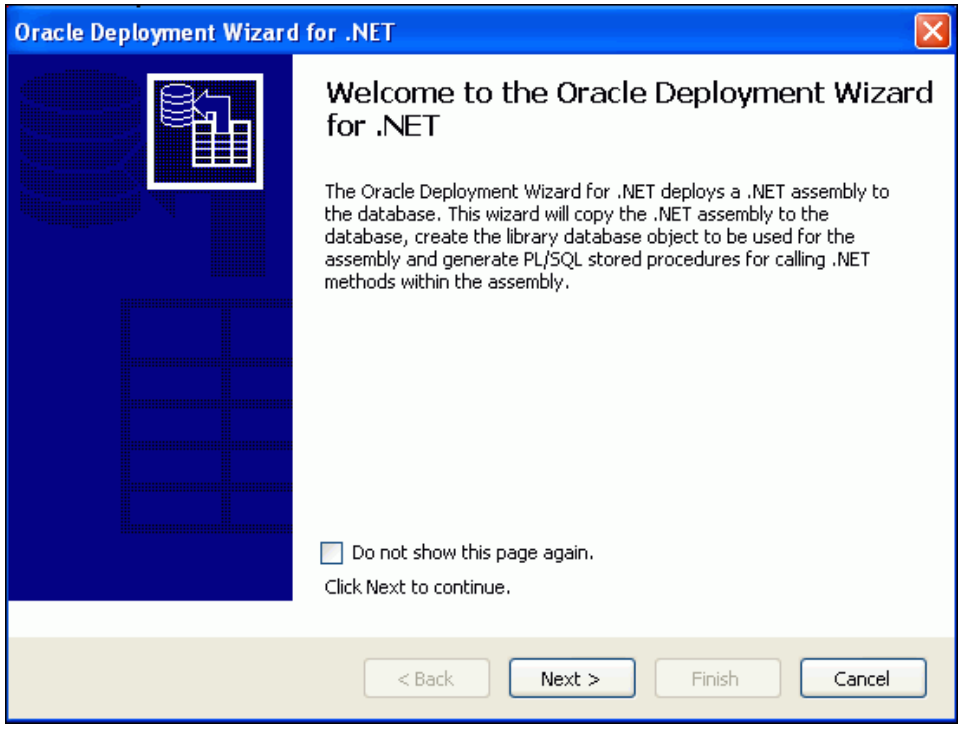
.NET ストアド・プロシージャをデプロイするには、次の手順を実行します。

1. 「**Build**」メニューから「**Deploy Solution**」を選択します。

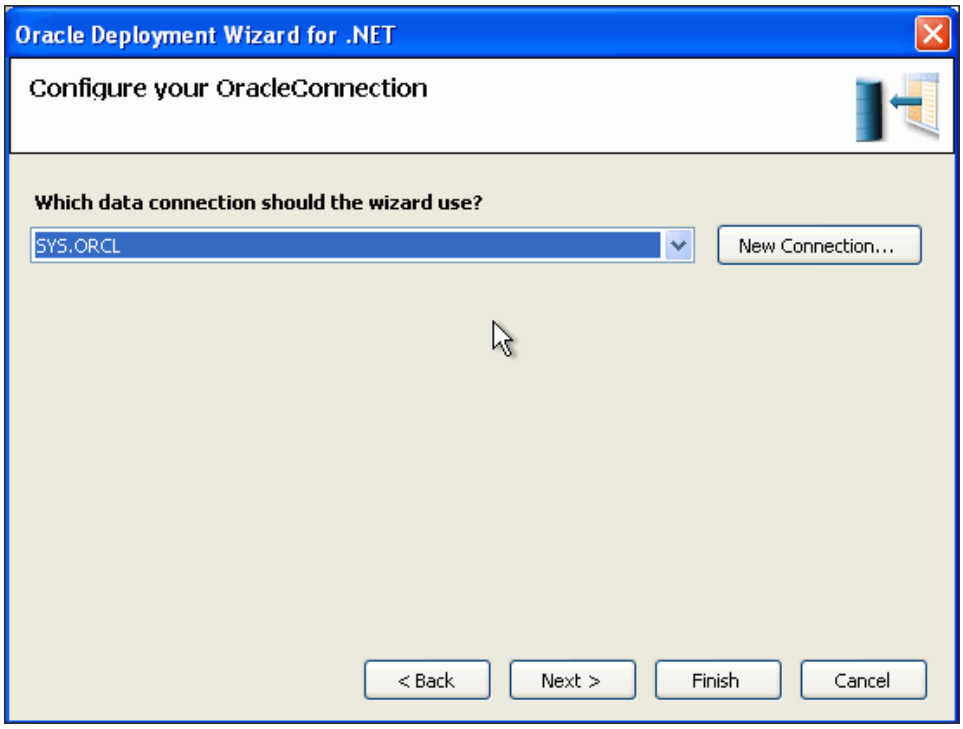


「.NET 用の Oracle デプロイメント・ウィザード」ウィンドウが表示されます。

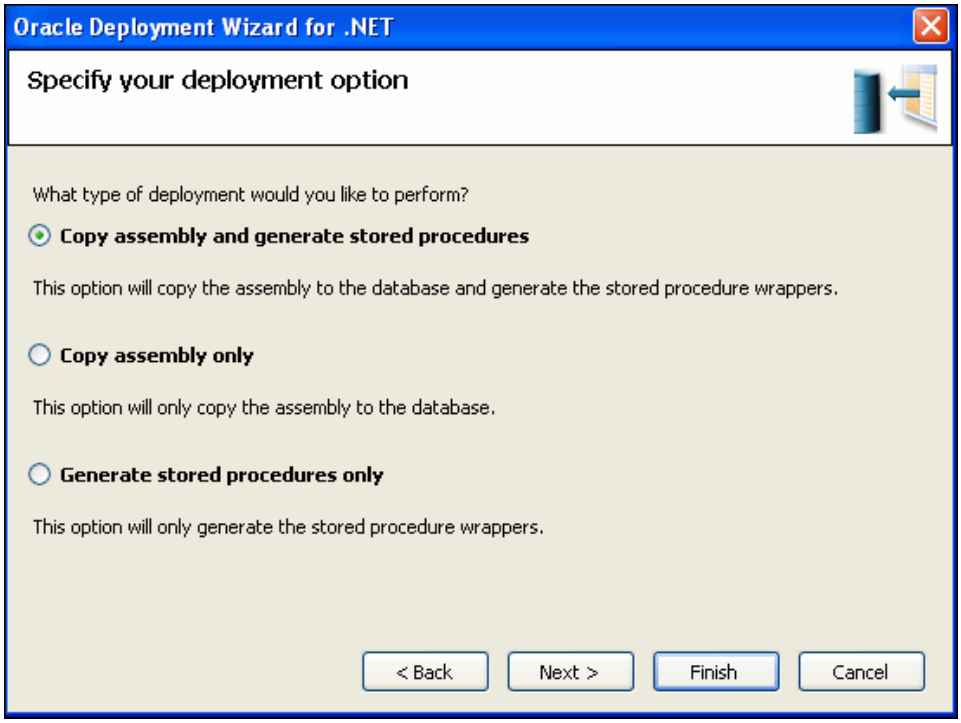
2. 「.NET 用の Oracle デプロイメント・ウィザード」ウィンドウで「**次へ**」をクリックします。



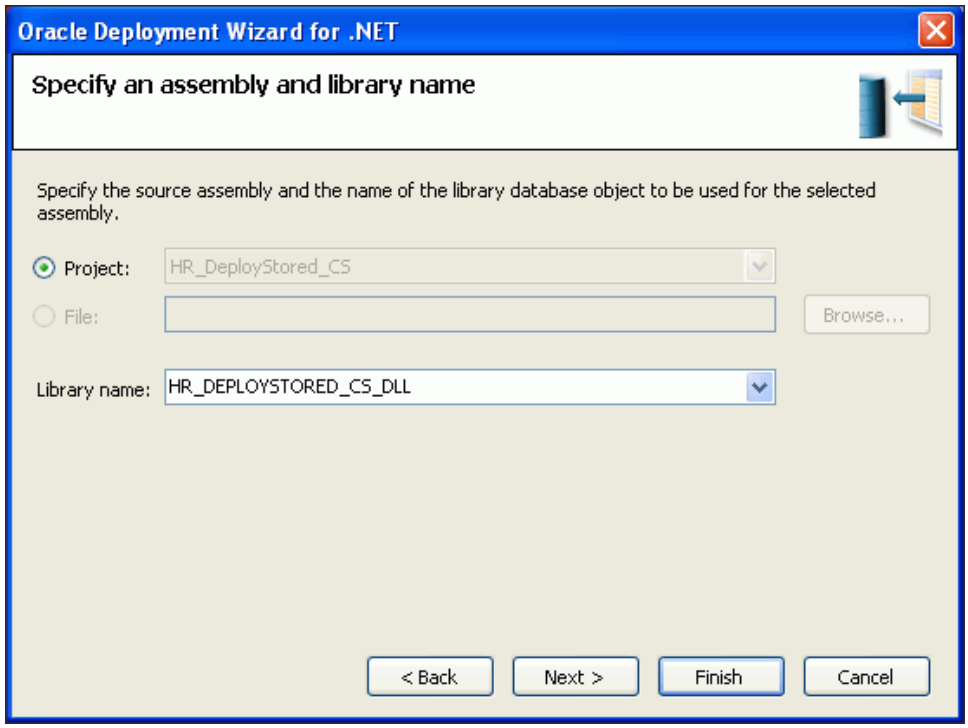
3. 「**OracleConnection を構成します**」ウィンドウで「**次へ**」をクリックします。



4. 「デプロイメント・オプションを指定します」ウィンドウで、最初のオプション（「**アセンブリをコピーしてストアド・プロシージャを生成**」）が選択されていることを確認し、「**次へ**」をクリックします。



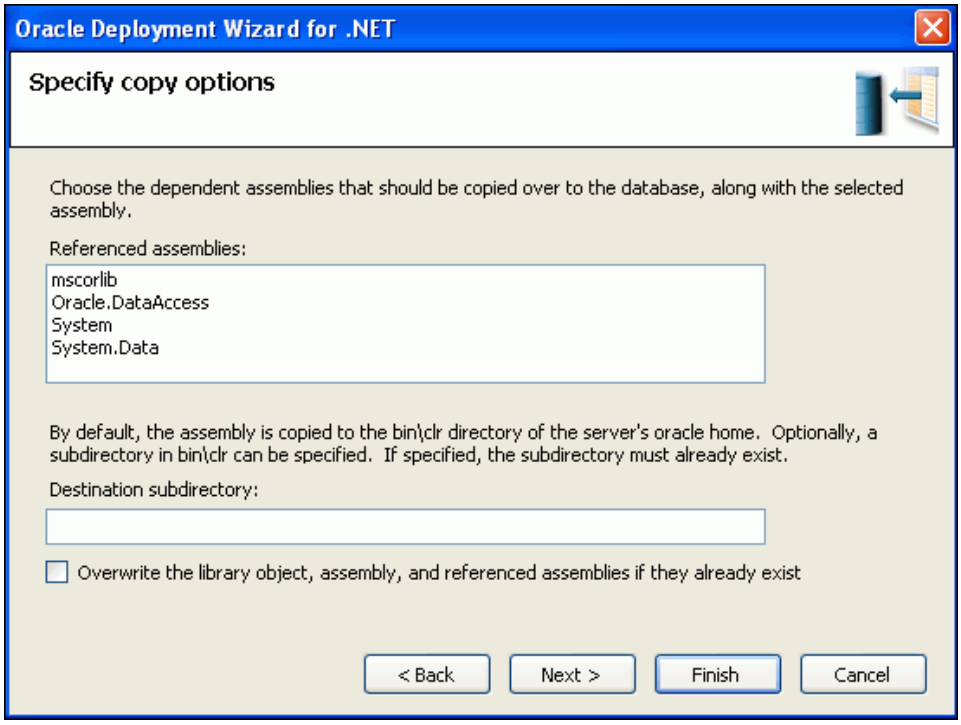
5. 「アセンブリおよびライブラリ名を指定します」ウィンドウでデフォルトを受け入れ、「**次へ**」をクリックします。



6. 「コピー・オプションの指定」ウィンドウでデフォルトを受け入れ、**「次へ」**をクリックします。

Visual Basic:

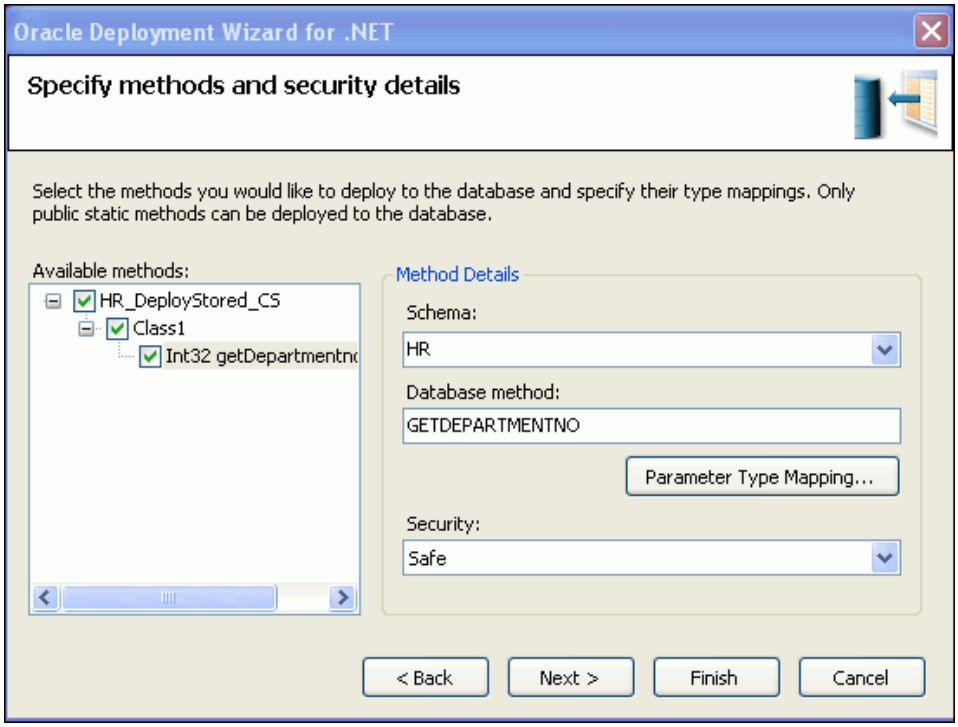
Visual Basic を使用している場合は、`Microsoft.VisualBasic` アセンブリも参照アセンブリとして表示されます。



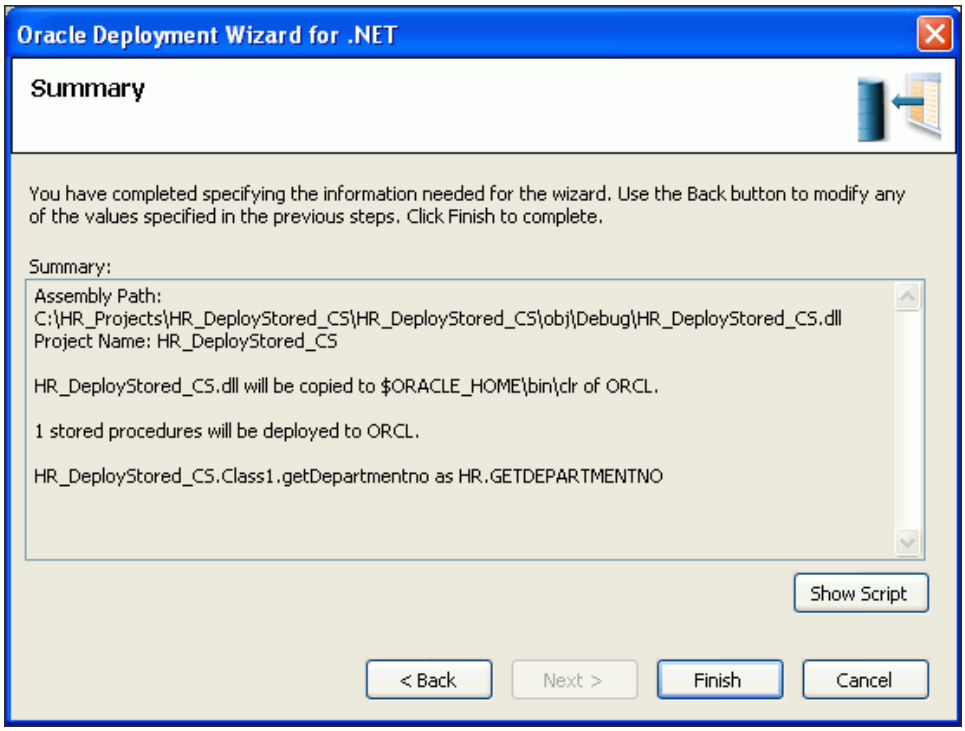
7. 「メソッドおよびセキュリティ詳細の指定」ウィンドウで、「使用可能なメソッド」にある「**HR_DeployStored_CS**」または「**HR_DeployStored_VB**」を開き、「**Class1**」を開いて「**`getDepartmentno()`**」メソッドを選択します。

   「メソッドの詳細」で、スキーマ・リストから「**HR**」を選択します。

   「**次へ**」をクリックします。



8. 「サマリー」ウィンドウで**「終了」**をクリックします。

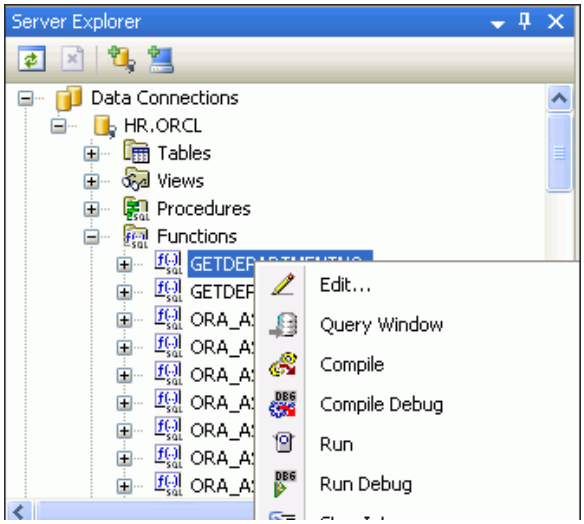


.NET ストアド・ファンクションおよびプロシージャの実行

これで、前の項でデプロイした .NET ストアド・プロシージャを実行する準備ができました。

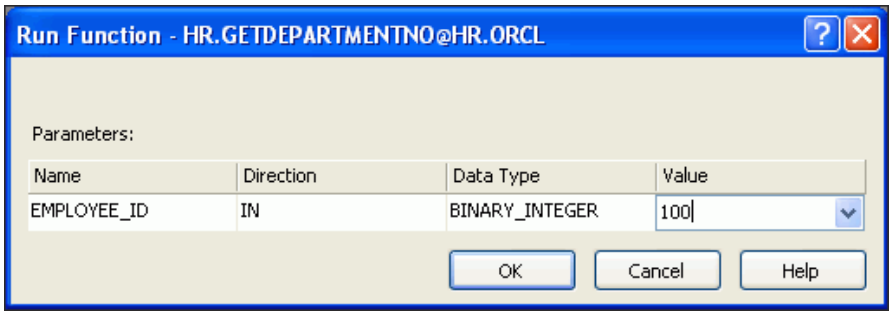
.NET ストアド・プロシージャを実行するには、次の手順を実行します。

1. Server Explorer で、HR.ORCL 接続をオープンします。「Functions」を開きます。「GETDEPARTMENTNO」を右クリックして**「Run」**を選択します。

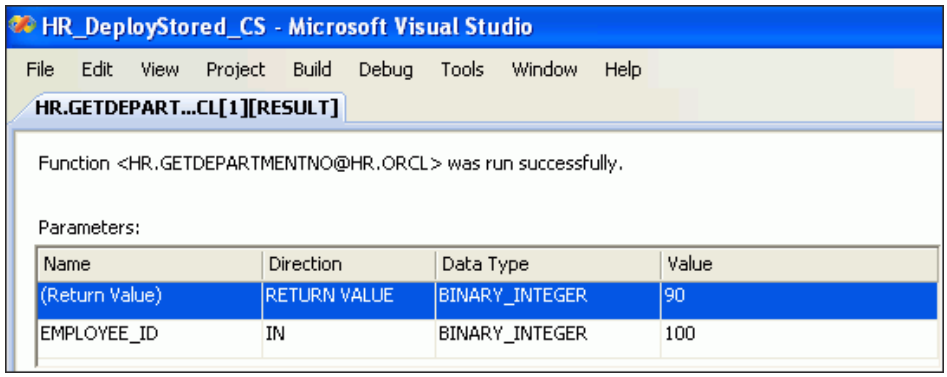


「Run Function」ウィンドウが表示されます。

2. 「Run Function」ウィンドウで、「EMPLOYEE_ID」の「Value」に 100 を入力します。**「OK」**をクリックします。



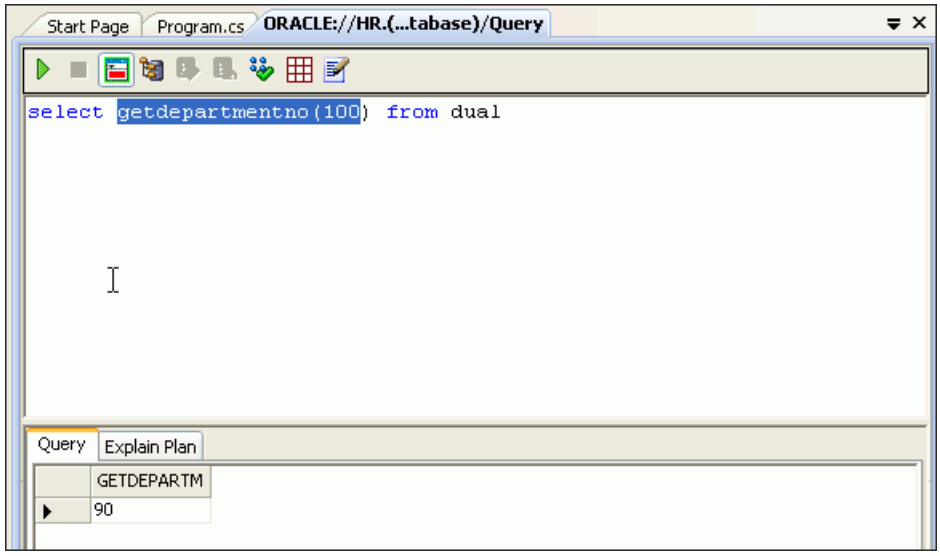
3. 部門の戻り値が 90 であることを確認します。これは、EMPLOYEE_ID 100 は部門 90 に含まれると示します。



問合せウィンドウでの .NET ストアド・プロシージャの実行

作成したばかりの .NET ストアド・プロシージャは、Server Explorer から実行する他に、ODT 問合せウィンドウを使用して実行することもできます。

1. Server Explorer で HR.ORCL スキーマを開きます。
2. 「Functions」を開き、「GETDEPARTMENTNO」を選択します。
3. 右クリックして「Query Window」を選択します。
4. Select getdepartmentno(100) from dual と入力します。
5. ツールバーから「Execute」をクリックします。



9

グローバリゼーション・サポートの組込み

この章の内容は次のとおりです。

参照 :

- 『Oracle Data Provider for .NET 開発者ガイド』の Oracle Data Provider for .NET グローバリゼーション・クラスに関する項
- 『Oracle Database 2 日で開発者ガイド』のグローバル環境での作業に関する項
- Microsoft 社の .NET 国際化対応に関するインターネット・サイト (`http://msdn.microsoft.com/en-us/goglobal/bb688096.aspx`)

グローバル・アプリケーションの概要

この章では、.NET での Oracle Database を使用するグローバル・アプリケーションの開発について説明します。ここでは、ロケール認識の開発やユーザーのロケールによる文化的な表記規則を使用したデータ表示など、グローバルにデプロイできるアプリケーションを開発するための基本的なタスクを扱います。また、Oracle Data Provider for .NET で使用できるグローバリゼーション・サポート機能についても説明します。

様々なロケールをサポートするグローバル対応アプリケーションを作成するには、正しい開発手順が必要です。

ロケールとは、各国語およびその言語が話されている地域を意味します。アプリケーション自体で、ユーザーのロケール・プリファレンスを認識し、ユーザーが期待する文化的な表記規則に従ってコンテンツを表示できる必要があります。正しい日付書式や数値書式など、該当するロケールの特性を使用してデータを表示することが重要です。Oracle Database は完全に国際化されており、グローバル・アプリケーションの開発およびデプロイに対応したグローバル・プラットフォームを提供しています。

.NET Framework を使用したグローバル・アプリケーションの開発

グローバル対応アプリケーションを計画する際には、次の 2 つの主なタスクを考える必要があります。

- **グローバリゼーション**は、様々な文化に適応できるアプリケーションを設計するプロセスです。
- **ローカリゼーション**は、特定の文化に合わせてリソースを翻訳するプロセスです。

.NET Framework の `System.Globalization` 名前空間には、言語、国および地域、カレンダ、日付の書式パターン、通貨、数値および文字列のソート順序など、文化に関連する情報を定義するクラスが含まれています。これらのクラスによりグローバル対応アプリケーションの開発プロセスが簡略化されます。そのため、ユーザーの文化を表す `CultureInfo` オブジェクトを `System.Globalization` 名前空間に含まれるメソッドに渡すと、一連の正しいルールとデータが適用されます。

.NET Framework ではリソースの作成とローカリゼーションもサポートされており、リソースをパッケージ化したりデプロイするためのモデルが提供されています。アプリケーションのリソースを特定の文化用にローカライズすることで、翻訳版のアプリケーションの開発がサポートされます。.NET Framework の基本クラス・ライブラリでは、アプリケーション・リソースを構築および操作するための様々なクラスを `System.Resources` 名前空間で提供しています。

ユーザーの正しいローカル規則でのデータ表示

アプリケーションのデータはユーザーが期待するとおりに表示される必要があり、そうでない場合は意味が誤って解釈される可能性があります。たとえば、12/11/05 は、アメリカ合衆国では 2005 年 12 月 11 日を意味しますが、イギリスでは 2005 年 11 月 12 日を意味します。数値書式と通貨書式にも同様の違いがあります。たとえば、ピリオド（.）は、アメリカ合衆国では小数点セパレータですが、欧州では千単位のセパレータです。

各言語には、それぞれ独自のソート規則があります。アルファベットの文字の順序に従う言語や、文字の画数に従う言語、単語の発音に従う言語などがあります。ユーザーが慣れている言語順序でソートされていないデータを表示すると、情報の検索が難しくなり、時間がかかることがあります。

アプリケーション・ロジックおよびデータベースから取得されるデータの量によっては、アプリケーション・レベルよりもデータベース・レベルでデータを書式設定した方が適切な場合があります。Oracle Database には、ユーザーのロケール・プリファレンスがわかっている場合に、データの表現を調整する機能が多く用意されています。

SQL*Plus への接続

次の例のいくつかでは、SQL*Plus を使用して、SYS や SYSTEM などのデータベース管理者権限を持つユーザーとして接続する必要があります。

> **参照：** 詳細は、『Oracle Database 2 日でデータベース管理者』のユーザー・アカウントのロックおよびロック解除に関する項を参照してください。

Oracle の日付書式の使用

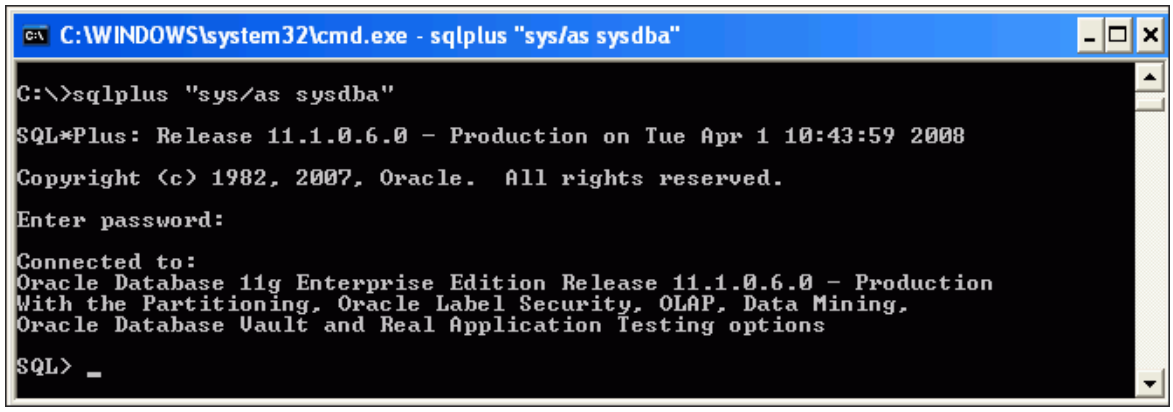
Oracle Database の日付表示書式には、標準の書式、短い書式、長い書式の 3 種類があります。次の手順では、アメリカ合衆国とドイツでの短い日付書式と長い日付書式の違いを示します。

Oracle の日付書式を変更するには、次の手順を実行します。

1. Windows のコマンド・プロンプトから次のコマンドを入力します。

```
C:¥>sqlplus "sys as sysdba"
Enter password:passwd
```

passwd は、データベースのインストール時に設定した Sys パスワードです。文字を入力しても、パスワードは表示されません。



2. SQL プロンプトで次のコマンドを入力します。

```
SQL> ALTER SESSION SET NLS_TERRITORY=america NLS_LANGUAGE=american;
```

「セッションが変更されました。」というメッセージが表示されます。

パラメータを現行の設定に設定しても問題はありません。セキュリティ上必要な場合に行うことはできます。現行の設定を確認するには、次のように入力します。

```
SQL> select * from v$nls_parameters;
```

または

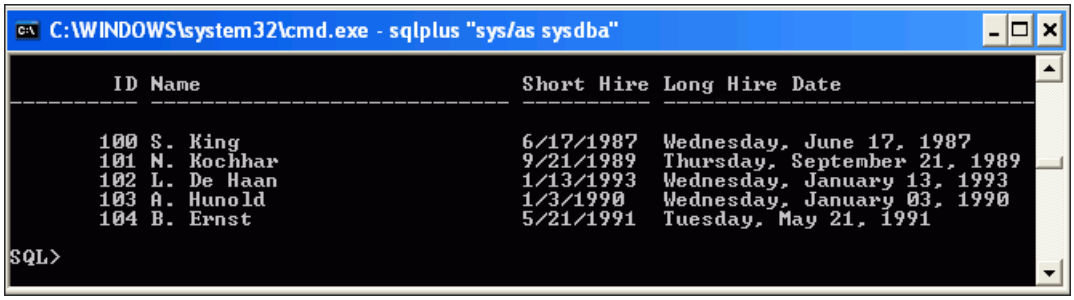
```
select * from v$nls_parameters where parameter = 'NLS_LANGUAGE';
```

3. SQL プロンプトで次の問合せを入力します。

```
SQL> SELECT employee_id "ID",
  SUBSTR (first_name,1,1)||'. '||last_name "Name",
  TO_CHAR (hire_date, 'DS') "Short Hire",
  TO_CHAR (hire_date, 'DL') "Long Hire Date"
  FROM hr.employees
  WHERE employee_id < 105;
```

現在は hr ではなく sys でログインしているため、hr スキーマの employees 表にアクセスするには、hr.employees を使用する必要があることに注意してください。

問合せ結果は、手順 1 で指定した米語書式で戻されます。



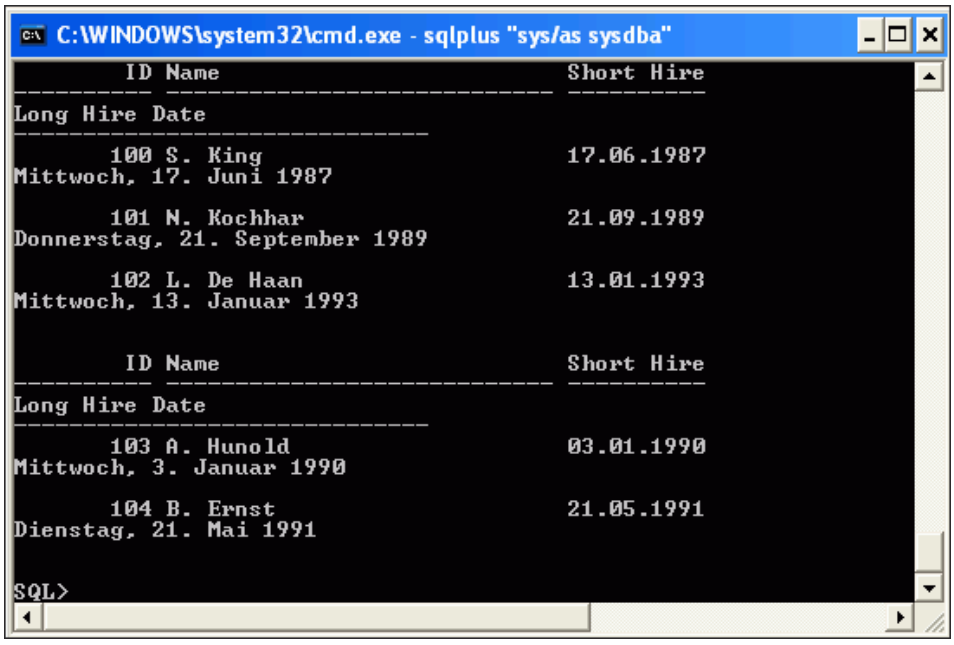
4. SQL プロンプトで次のコマンドを入力します。

```
SQL> ALTER SESSION SET NLS_TERRITORY=germany NLS_LANGUAGE=german;
```

「セッションが変更されました。」というメッセージが表示されます。

5. SQL プロンプトで、手順 3 の問合せを入力します。

問合せ結果は、手順 4 で指定したドイツ語書式で戻されます。



Oracle の数値書式の使用

小数点文字およびグループ・セパレータにも違いがあります。次の手順では、アメリカ合衆国とドイツでのこれらの違いを示します。

Oracle の数値書式を変更するには、次の手順を実行します。

1. SQL プロンプトで次のコマンドを入力します。

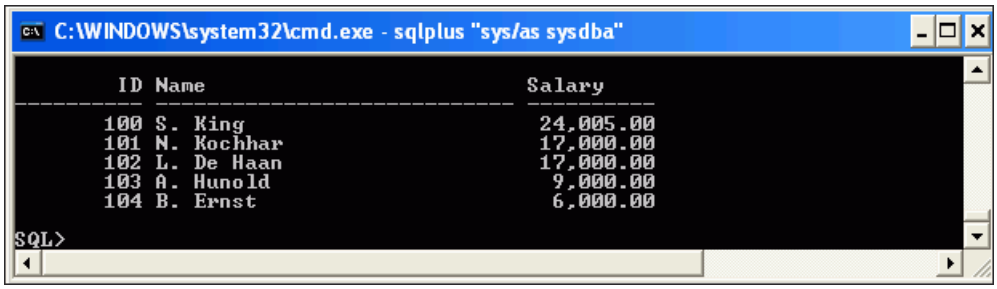
```
SQL> ALTER SESSION SET NLS_TERRITORY=america NLS_LANGUAGE=american;
```

「セッションが変更されました。」というメッセージが表示されます。

2. SQL プロンプトで次の問合せを入力します。

```
SQL> SELECT employee_id "ID",
  SUBSTR (first_name,1,1)||'. '||last_name "Name",
  TO_CHAR (salary, '99G999D99') "Salary"
  FROM hr.employees
  WHERE employee_id < 105;
```

問合せ結果は、手順 1 で指定した米語書式で戻されます。



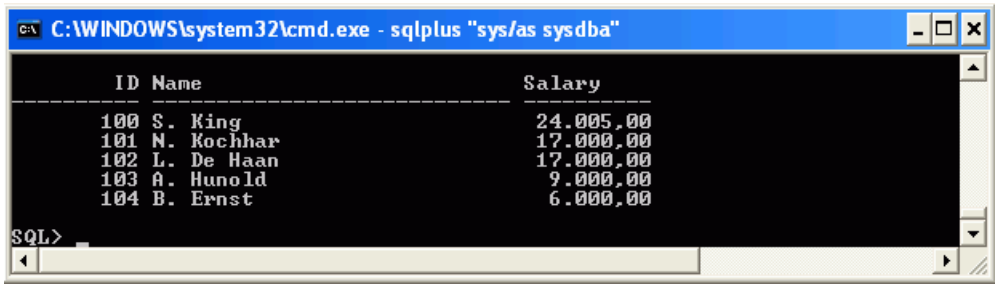
3. SQL プロンプトで次のコマンドを入力します。

```
SQL> ALTER SESSION SET NLS_TERRITORY=germany;
```

「セッションが変更されました。」というメッセージが表示されます。

4. SQL プロンプトで、手順 2 の問合せを入力します。

問合せ結果は、手順 3 で指定したドイツ語書式で戻されます。



Oracle の言語ソートの使用

スペインでは、伝統的に ch、ll および ñ がスペイン独自の文字として扱われ、それぞれ c、l および n の後に順序付けられています。次の手順では、Chen、Chung および Colmenares という従業員名にスペイン語のソートを使用した場合の結果を説明します。

Oracle の言語ソートを変更するには、次の手順を実行します。

1. SQL プロンプトで次のコマンドを入力します。

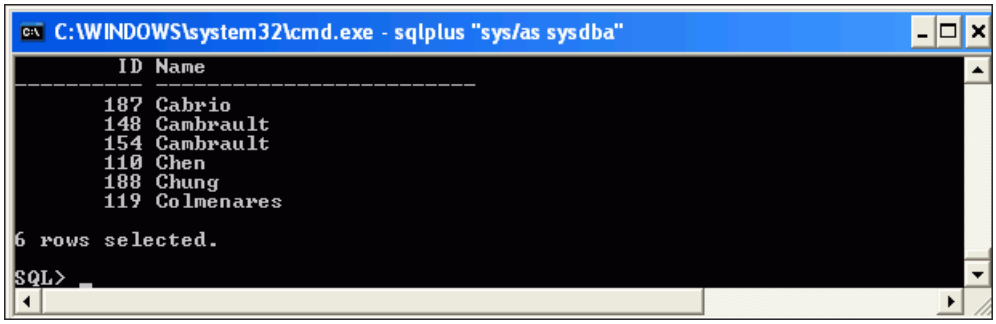
```
SQL> ALTER SESSION SET NLS_SORT=binary;
```

「セッションが変更されました。」というメッセージが表示されます。

2. SQL プロンプトで次の問合せを入力します。

```
SQL> SELECT employee_id "ID",
          last_name "Name"
  FROM hr.employees
  WHERE last_name LIKE 'C%'
  ORDER BY last_name;
```

問合せ結果は、手順 1 で指定したバイナリ・ソート順で戻されます。

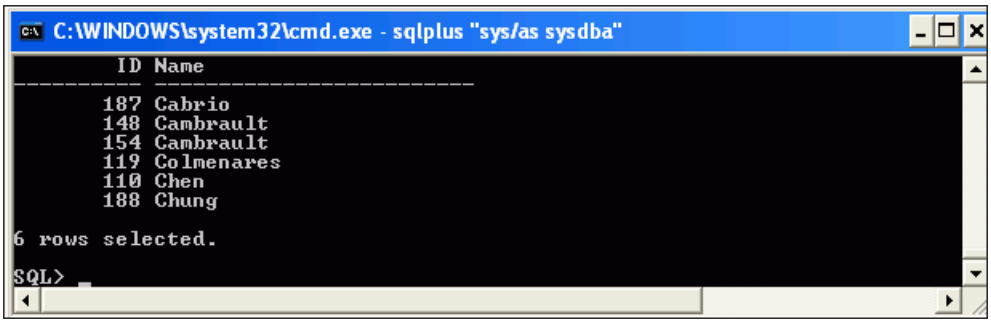


3. SQL プロンプトで次のコマンドを入力します。

```
SQL> ALTER SESSION SET NLS_SORT=spanish_m;
```

「セッションが変更されました。」というメッセージが表示されます。

4. SQL プロンプトで、手順 2 の問合せを入力します。
5. 問合せ結果は、手順 3 で指定したスペイン語のソート順で戻されます。



Oracle のエラー・メッセージ

NLS_LANGUAGE パラメータでは、データベースのエラー・メッセージの言語を制御することもできます。次の手順で示すとおり、SQL 問合せを実行する前にこのパラメータを設定することで、ローカル言語固有のエラー・メッセージが戻されるようになります。

Oracle の NLS 言語パラメータを変更するには、次の手順を実行します。

1. SQL プロンプトで次のコマンドを入力します。

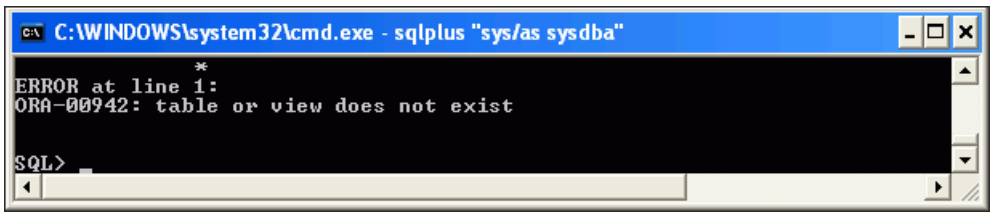
   ```
   SQL> ALTER SESSION SET NLS_LANGUAGE=american;
   ```

   「セッションが変更されました。」というメッセージが表示されます。

2. SQL プロンプトで次の問合せを入力します。

   ```
   SQL> SELECT * FROM managers;
   ```

   問合せ結果として、手順 1 で指定した言語でエラー・メッセージが戻されます。



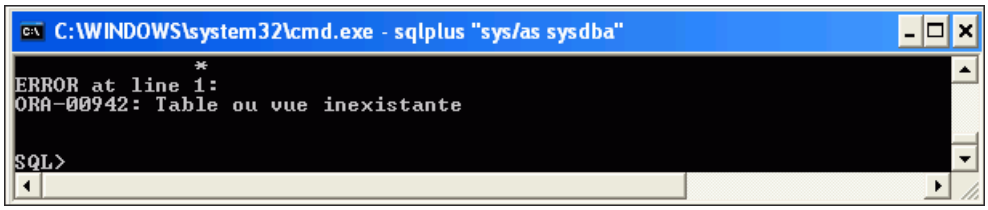
3. SQL プロンプトで次のコマンドを入力します。

   ```
   SQL> ALTER SESSION SET NLS_LANGUAGE=french;
   ```

   「セッションが変更されました。」というメッセージが表示されます。

4. SQL プロンプトで、手順 2 の問合せを入力します。

   問合せ結果として、手順 3 で指定した言語でエラー・メッセージが戻されます。



5. 言語、ローカルおよびソート設定をリセットし、元の値に戻します。

.NET と Oracle Database のロケール環境の同期化

グローバル・アプリケーションを開発するときは、データベースとクライアントのユーザー・ロケール設定を常に同期させます。同期させない場合は、文化によって異なる情報が矛盾した状態でアプリケーションから表示されることがあります。たとえば、.NET アプリケーションでは、SQL 操作を実行する前に、アプリケーション・ユーザーの Culture ID を正しい NLS_LANGUAGE パラメータ値および NLS_TERRITORY パラメータ値にマップする必要があります。

表 9-1 に、一般的に使用されるロケールの .NET 環境と Oracle 環境での定義を示します。

表 9-1 一般的な NLS_LANGUAGE および NLS_TERRITORY パラメータ

Culture	Culture ID	NLS_LANGUAGE	NLS_TERRITORY
中国語（中華人民共和国）	zh-CN	SIMPLIFIED CHINESE	CHINA
中国語（台湾）	zh-TW	TRADITIONAL CHINESE	TAIWAN
英語（アメリカ合衆国）	en-US	AMERICAN	AMERICA
英語（イギリス）	en-GB	ENGLISH	UNITED KINGDOM
フランス語（カナダ）	fr-CA	CANADIAN FRENCH	CANADA
フランス語（フランス）	fr-FR	FRENCH	FRANCE
ドイツ語	de	GERMAN	GERMANY
イタリア語	it	ITALIAN	ITALY
日本語	ja	JAPANESE	JAPAN
韓国語	ko	KOREAN	KOREA
ポルトガル語（ブラジル）	pt-BR	BRAZILIAN PORTUGUESE	BRAZIL
ポルトガル語	pt	PORTUGUESE	PORTUGAL
スペイン語	es	SPANISH	SPAIN

Oracle Data Provider for .NET でのクライアント・グローバリゼーションのサポート

Oracle Data Provider for .NET を使用すると、Oracle Database で定義されている文化によって異なる表記規則を使用して、適切な文字列書式、日付、時間、金額、数値、ソート順序を保持したり、カレンダをサポートするなど、文化によって異なるデータをアプリケーションで操作できます。デフォルトのグローバリゼーション設定は、クライアントの NLS_LANG パラメータによって決まります。このパラメータは、ローカル・コンピュータの Windows レジストリで定義されます。OracleConnection Open メソッドにより接続が確立されるときに、NLS_LANG パラメータの値で指定されたグローバリゼーション・パラメータを使用してセッションが暗黙的にオープンされます。

クライアント・グローバリゼーション設定

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。OracleGlobalization オブジェクト・プロパティを変更しても、セッションまたはスレッドのグローバリゼーション設定は変更されません。次の項では、セッションおよびスレッド・レベルでグローバリゼーション設定を変更する方法を説明します。

.NET アプリケーションでは、OracleGlobalization.GetClientInfo() 静的メソッドをコールすることでグローバリゼーション設定を取得できます。次の OracleGlobalization のサンプル・コードは、.NET の値の一部を取得する方法を示しています。

Visual C#:

```
using System;
using Oracle.DataAccess.Client;

class ClientGlobalizationSample
{
  static void Main()
  {
    OracleGlobalization ClientGlob = OracleGlobalization.GetClientInfo();
    Console.WriteLine("Client machine language: " + ClientGlob.Language);
    Console.WriteLine("Client characterset: " + ClientGlob.ClientCharacterSet);
  }
}
```

Visual Basic:

```
Imports System
Imports Oracle.DataAccess.Client

Class ClientGlobalizationSample
  Shared Sub Main()
    Dim ClientGlob As OracleGlobalization = OracleGlobalization.GetClientInfo()
    Console.WriteLine("Client machine language: " + ClientGlob.Language)
    Console.WriteLine("Client characterset: " + ClientGlob.ClientCharacterSet)
  End Sub
End Class
```

セッション・グローバリゼーション設定の使用

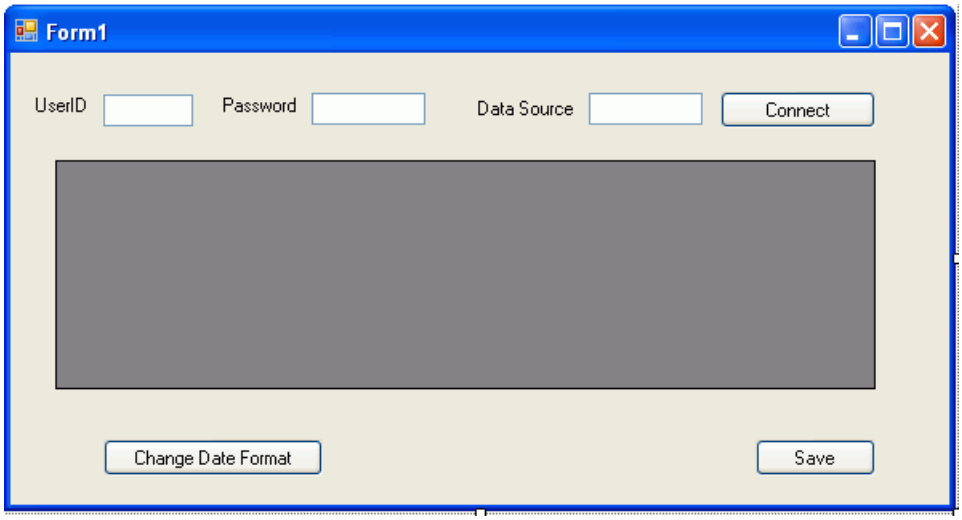
セッション・グローバリゼーション・パラメータは、最初はクライアント・グローバリゼーション設定と同じですが、変更が可能です。セッション・パラメータを変更するには、データベース接続を確立する必要があります。その後、`OracleConnection` オブジェクトの `GetSessionInfo()` メソッドをコールしてセッション・グローバリゼーション設定を取得します。次に、必要に応じてグローバリゼーション設定を変更し、`SetSessionInfo(OracleGlobalization)` メソッドを使用して `OracleConnection` オブジェクトに設定を保存します。

グローバリゼーション・セッション設定を指定するには、次の手順を実行します。

1. アプリケーション `HR_Connect_CS` または `HR_Connect_VB` を開きます。
2. 付録 B「フォームのコピー」の手順に従って、第 4 章の最後で完成させた `Form3.xx` のコピーを作成し、`Form5.xx` という名前を付けます。
3. プロジェクトの **Form1** を開き、設計ビューに切り替えます。
4. **「View」**メニューから**「Toolbox」**を選択します。
5. 「Toolbox」の「Windows Forms」の下にある**「Button」**を Form1 にドラッグ・アンド・ドロップします。
6. 新しい**「Button」**を右クリックして**「Properties」**を選択します。「Properties」ウィンドウが表示されます。

7. 「Properties」ウィンドウで、次のプロパティを設定します。
   - 「Appearance」の下で、**「Text」**を `Change Date Format` に変更します。
   - 「Design」の下で、**「(Name)」**を `date_change` に変更します。

   Form1 の外観は次のようになります。



「Properties」ウィンドウで**「Events」**（稲妻のアイコン）をクリックすると、`date_change_Click()` が日付ボタンの「Event」として表示されます。

8. 作成したばかりの新しい `date_change_Click()` メソッドを開き、次のコードを追加して日付書式を標準の `DD-MON-RR` から `YYYY-MM-DD` に変更し、`DataSet` を更新します。

Visual C#:

```
si.DateFormat = "YYYY-MM-DD";
conn.SetSessionInfo(si);

ds.Clear();
da.Fill(ds);
departments.DataSource = ds.Tables[0];
```

Visual Basic:

```
si.DateFormat = "YYYY-MM-DD"
conn.SetSessionInfo(si)

ds.Clear()
da.Fill(ds)
departments.DataSource = ds.Tables(0)
```

`ds.Clear()` をコールすると、変更したデータがポストされる前に古い結果が消去されることに注意してください。

また、`si` クラス変数の宣言およびセッション・グローバリゼーション情報の取得は、手順 10 および手順 11 で行います。

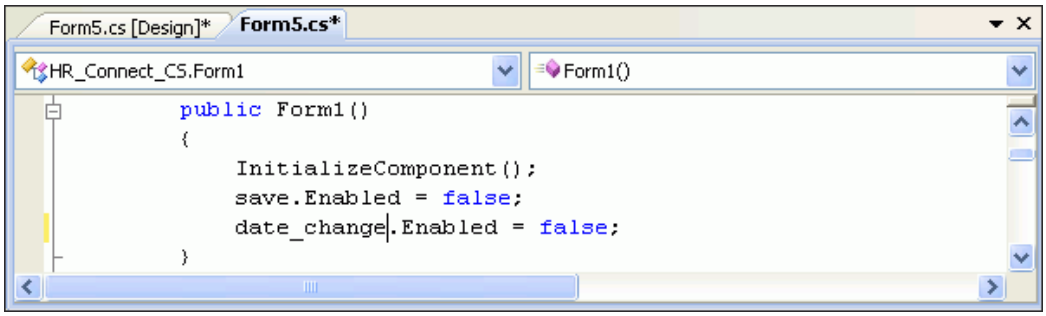
9. 該当するメソッド内に、次のコードを追加します。

Visual C#: `Form1()` メソッドに追加

```
date_change.Enabled = false;
```

Visual Basic: `Form1_Load` メソッドに追加

```
date_change.Enabled = false
```



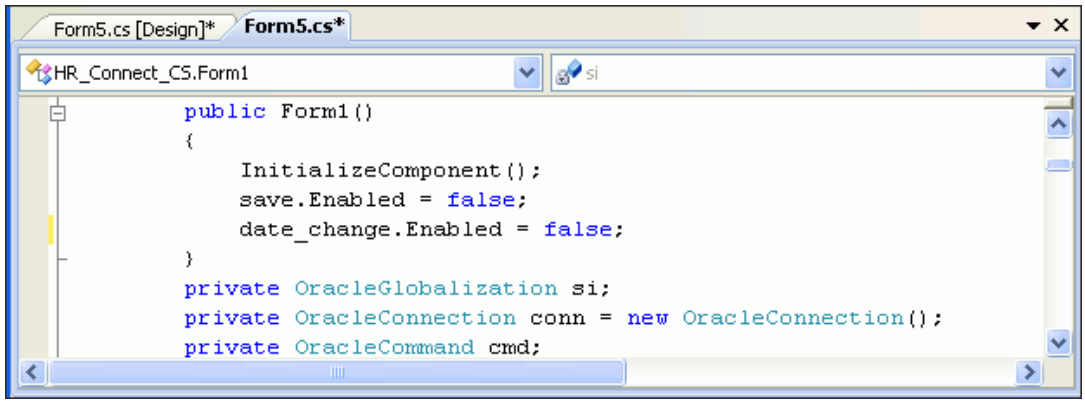
10. 次のコードを使用して、既存の Form1 クラス宣言の `public Form1()` ブロックの直後に次のクラス変数を追加します。

Visual C#:

```
private OracleGlobalization si;
```

Visual Basic:

```
private si As OracleGlobalization
```



11. connect_Click() メソッドの Try ブロックに、次の操作を実行するコードを追加します。

    - OracleGlobalization オブジェクトの値の取得
    - EMPLOYEES 表からのデータの取得（新しい問合せ）
    - 「**Change Date Format**」ボタンの有効化

    変更したコードは太字で示しています。

    Visual C#:

```
conn.Open();
connect.Enabled = false;

si = conn.GetSessionInfo();

string sql = "select employee_id, first_name, last_name, TO_CHAR(hire_date)" +
   " ¥"Hire Date¥" from employees where employee_id < 105";
cmd = new OracleCommand(sql, conn);
cmd.CommandType = CommandType.Text;

da = new OracleDataAdapter(cmd);
cb = new OracleCommandBuilder(da);
ds = new DataSet();

da.Fill(ds);

departments.DataSource = ds.Tables[0];

save.Enabled = true;
date_change.Enabled = true;
```

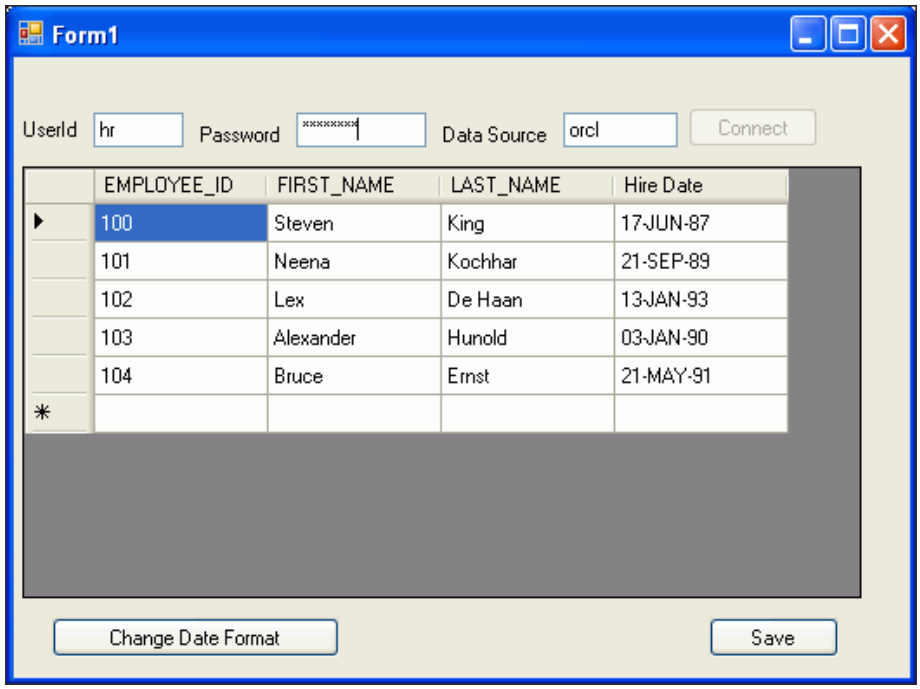
    Visual Basic:

```
conn.Open()
connect.Enabled = false

si = conn.GetSessionInfo()

Dim sql As String = "select employee_id, first_name, last_name, " & _
    "TO_CHAR(hire_date) ""Hire Date"" from employees where employee_id < 105"
cmd = new OracleCommand(sql, conn)
cmd.CommandType = CommandType.Text

da = new OracleDataAdapter(cmd)
cb = new OracleCommandBuilder(da)
ds = new DataSet()

da.Fill(ds)

departments.DataSource = ds.Tables[0]

save.Enabled = true
date_change.Enabled = true
```

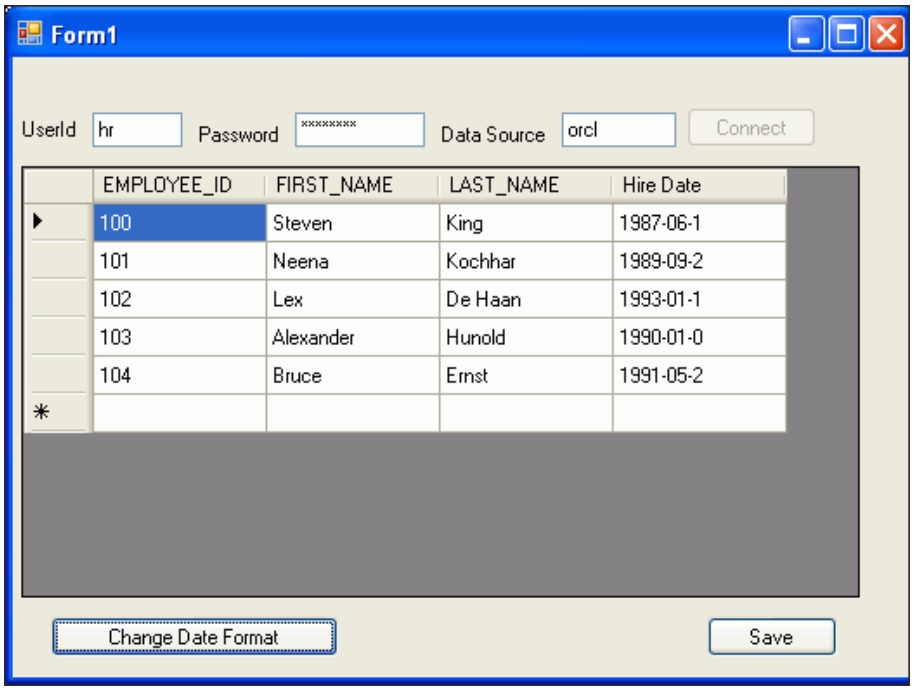
12. Form1 を保存します。

13. **[F5]** キーボード・ショートカットを使用してアプリケーションを実行します。

    アプリケーションがデータベースに正しく接続されると、データ・グリッドに問合せ結果が移入されます。



14. 「**Change Date Format**」をクリックします。



    日付書式が元の `DD-MON-RR` から `YYYY-MM-DD` に変更されることを確認します。

15. アプリケーションを閉じます。

スレッドベースのグローバリゼーション設定

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、プログラムで変更できます。ODP.NET 型と文字列との間で変換が行われるときに、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` クラスの `GetThreadInfo()` 静的メソッドをコールします。`SetThreadInfo()` 静的メソッドをコールすると、スレッドのグローバリゼーション設定が設定されます。

文化によって異なるデータを操作する場合、ODP.NET クラスおよび構造体は、`OracleGlobalization` 設定のみを使用します。.NET スレッドの文化情報は使用されません。アプリケーションで .NET 型のみを使用する場合は、`OracleGlobalization` の設定は影響しません。ただし、ODP.NET 型と .NET 型の間で変換を行う場合は、該当する箇所で `OracleGlobalization` の設定が使用されます。

注意： `System.Threading.Thread.CurrentThread.CurrentCulture` プロパティを変更しても、スレッドまたはセッションの `OracleGlobalization` 設定は変わりません。その逆も同様です。

A

Oracle Database インスタンスの起動および停止

データベースの停止や再起動は頻繁に必要になる操作です。

Oracle Database インスタンスを起動するには、次の手順を実行します。

1. **「スタート」**ボタンから、**「プログラム」**、**「管理ツール」**、**「サービス」**の順に選択し、**OracleService*DatabaseName*** を選択します。DatabaseName は `tnsnames.ora` ファイルで示されているデータベースの `service_name` です。詳細は、2-10 ページの「NET 接続の別名の構成」を参照してください。
2. 左側のパネルで、リンクをクリックして、サービスを開始します。
3. データベースのサービスが開始され、**「データベースの起動」**ウィンドウが表示されます。「`OracleService service was started successfully`」というメッセージが表示されるまで、操作は行わないでください。

Oracle Database インスタンスを停止するには、次の手順を実行します。

1. **「スタート」**ボタンから、**「プログラム」**、**「管理ツール」**、**「サービス」**の順に選択し、**OracleService*DatabaseName*** を選択します。
2. 左側のパネルで、リンクをクリックして、サービスを停止します。
3. データベースの停止が開始され、**「データベースの停止」**ウィンドウが表示されます。「`OracleService service was stopped successfully`」というメッセージが表示されるまで、操作は行わないでください。

B

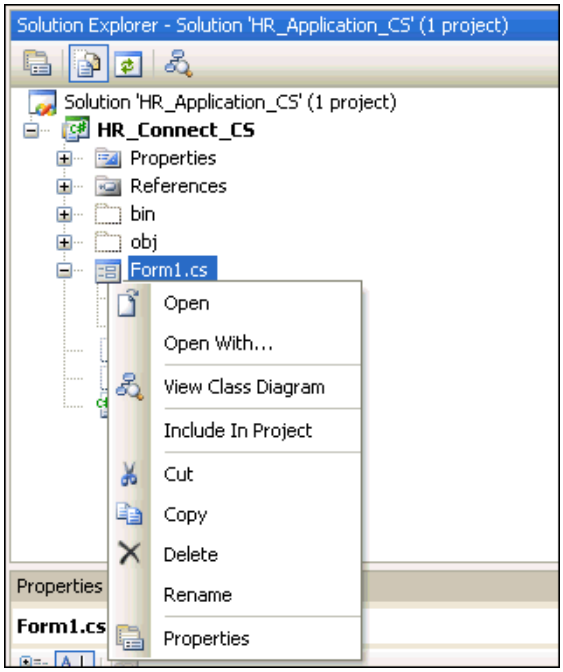
フォームのコピー

Oracle を使用したアプリケーション開発の様々な側面について学習するために、このアプリケーションを使用する際は、フォームをコピーして再利用することが必要になる場合があります。

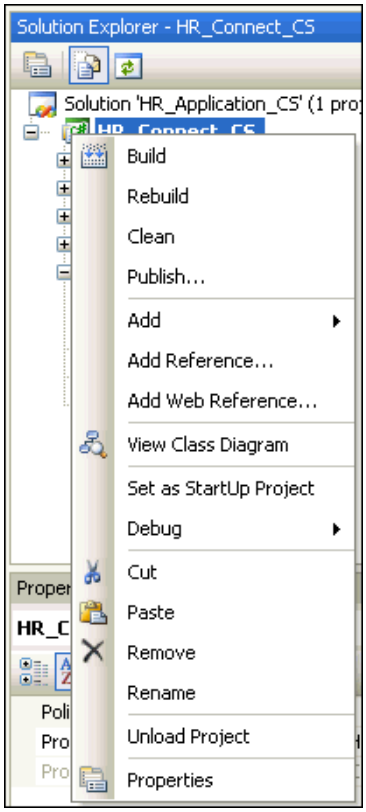
既存のフォームのコピーを作成するには、次の手順を実行します。

1. Solution Explorer で、コピーする必要のある `Form1.xx` またはその他のファイルを右クリックします。**「Copy」**を選択します。

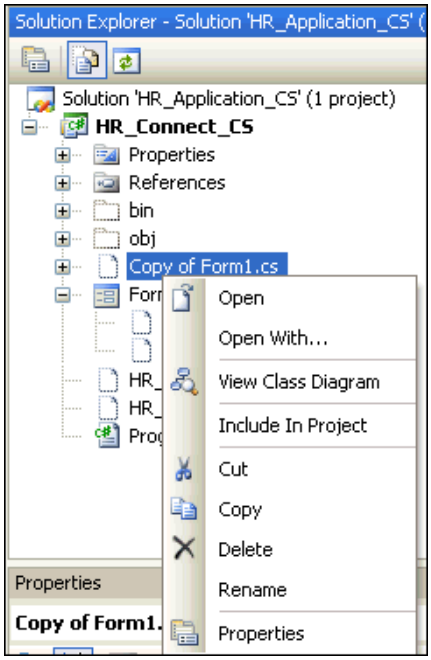
   `Form1.xx` が Solution Explorer に表示されない場合は、**「Project」**メニューから、**「Show All Files」**を選択します。



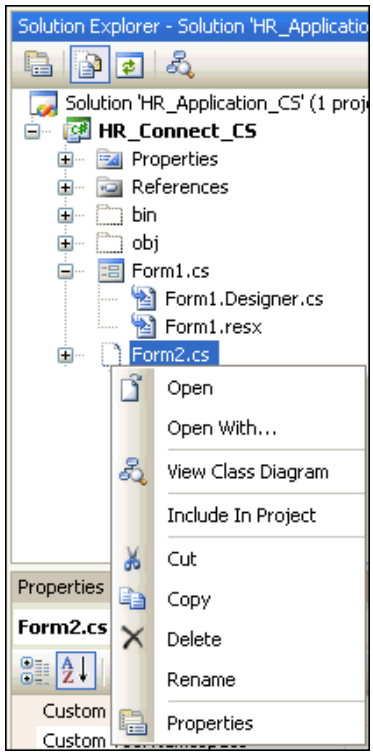
2. **HR_Connect_CS** またはその他のプロジェクトを右クリックします。「**Paste**」を選択します。



3. 「**Copy of Form1.cs**」を右クリックします。「**Rename**」を選択します。フォームの名前を **Form2.cs** に変更します。



4. 「Form2.cs」を右クリックして、**「Include In Project」**を選択します。



5. 「`Form1.cs`」を右クリックして、**「Exclude From Project」**を選択します。

これらの手順を逆に実行することで、プロジェクトにフォームを含めたり、プロジェクトからフォームを除外することができます。

注意： 通常、この処理は問題なく実行されます。問題が発生した場合は、**「Build」**メニューから**「Rebuild Solution」**の実行してみてください。

索引

O

P

R

S

T

U

V

W

あ

い

え

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

な

に

は

ひ

ふ

へ

ほ

め

ゆ

ら

れ

ろ