

## **Oracle® Database Express Edition**

2 Day Plus .NET Developer Guide

10g Release 2 (10.2)

**B25312-01**

February 2006

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Retek are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

---

---

# Contents

<b>Preface</b> .....	vii
Audience.....	vii
Documentation Accessibility .....	vii
Related Documents .....	vii
Conventions .....	viii
 <b>1 Introduction</b>	
What is Microsoft .NET Framework.....	1-1
Introduction to Oracle Data Provider for .NET .....	1-2
Introduction to Oracle Developer Tools for Visual Studio .NET .....	1-2
Introduction to .NET Stored Procedures.....	1-2
 <b>2 Installing Oracle Developer Tools</b>	
What You Need .....	2-1
Installing Oracle Developer Tools .....	2-2
Unlocking the User Account .....	2-4
Uninstalling Oracle Developer Tools .....	2-4
 <b>3 Connecting to the Database</b>	
Starting a New Project.....	3-1
Adding a Reference.....	3-2
Adding Initial Programmatic Statements.....	3-4
Adding Connection Elements.....	3-5
Using the Connection Object.....	3-11
Running the Application .....	3-14
Error Handling .....	3-14
Closing the Database Connection.....	3-18
 <b>4 Building an Oracle Data Provider for .NET Application</b>	
Copying a Project .....	4-1
Using the Command Object.....	4-3
Retrieving Data: a Simple Query .....	4-4
Retrieving Data: Bind Variables .....	4-5

Retrieving Data: Multiple Values .....	4-7
Using the DataSet Class with Oracle Data Provider for .NET .....	4-8
Inserting, Deleting and Updating Data .....	4-13
<b>5 Using Oracle Developer Tools for Visual Studio .NET</b>	
Connecting to the Oracle Database.....	5-1
Creating a Table and Its Columns .....	5-3
Creating a Table Index .....	5-5
Adding Table Constraints .....	5-6
Adding Data to a Table .....	5-8
Generating Code Automatically.....	5-9
Enabling Database Updates .....	5-13
<b>6 Using PL/SQL Stored Procedures and REF Cursors</b>	
Introduction to PL/SQL Packages and Package Bodies .....	6-1
Introduction to PL/SQL Stored Procedures.....	6-1
Introduction to Ref Cursors.....	6-1
Creating a PL/SQL Stored Procedure that Uses Ref Cursors.....	6-2
Running a PL/SQL Stored Procedure Using Oracle Data Provider for .NET .....	6-8
<b>7 Deploying .NET Stored Procedures</b>	
Starting the Common Language Runtime Service .....	7-1
Creating an Oracle Project.....	7-2
Creating a New Connection .....	7-3
Creating .NET Stored Functions and Procedures.....	7-5
Deploying .NET Stored Functions and Procedures.....	7-7
Running .NET Stored Functions and Procedures .....	7-11
<b>8 Including Globalization Support</b>	
Introduction to Global Applications.....	8-1
Developing Global Applications with the .NET Framework.....	8-1
Presenting Data in the Correct User Local Convention .....	8-2
Oracle Date Formats .....	8-2
Oracle Number Formats.....	8-3
Oracle Linguistic Sorts.....	8-4
Oracle Error Messages.....	8-5
Synchronizing the .NET and Oracle Database Locale Environments .....	8-6
Client Globalization Support in Oracle Data Provider for .NET .....	8-7
Client Globalization Settings .....	8-7
Session Globalization Settings.....	8-8
Thread-Based Globalization Settings .....	8-12

## **Index**

## List of Examples

3-1	Adding Initial Programmatic Statements: C#.....	3-5
3-2	Adding Initial Programmatic Statements: VB .....	3-5
3-3	Easy Connect Naming Method Syntax for Data Source .....	3-12
3-4	Creating an OracleConnection Object: C#.....	3-12
3-5	Creating an OracleConnection Object: VB .....	3-12
3-6	Building and Opening a Connection: C#.....	3-13
3-7	Building and Opening a Connection: VB .....	3-13
3-8	Disabling the Connect Button: C# .....	3-13
3-9	Disabling the Connect Button: VB .....	3-14
3-10	Error Handling with Try-Catch-Finally Syntax: C#.....	3-15
3-11	Error Handling with Try-Catch-Finally Syntax: VB .....	3-15
3-12	Catching Common Database Error Messages: C#.....	3-16
3-13	Catching Common Database Error Messages: VB .....	3-16
3-14	Closing and Disposing a Connection: C#.....	3-18
3-15	Closing and Disposing a Connection: VB .....	3-18
3-16	Closing and Disposing a Connection when Out of Scope: C# .....	3-18
4-1	Creating a SQL Statement String: C# .....	4-3
4-2	Creating a SQL Statement String: VB.....	4-3
4-3	Using a Command to Query the Database: C#.....	4-3
4-4	Using a Command to Query the Database: VB .....	4-3
4-5	Starting the OracleDataReader: C#.....	4-4
4-6	Starting the OracleDataReader: VB .....	4-4
4-7	Retrieving a Value: C#.....	4-5
4-8	Retrieving a Value: VB .....	4-5
4-9	SELECT Statement without Bind Variables .....	4-6
4-10	SELECT Statement with Bind Variables .....	4-6
4-11	Using a Bind Variable: C#.....	4-6
4-12	Using a Bind Variable: VB .....	4-7
4-13	UPDATE Statement with Bind Variables .....	4-7
4-14	Querying for a Multiple Column Multiple Row Result .....	4-8
4-15	Looping Through a Multi-Row Query Result: C# .....	4-8
4-16	Looping Through a Multi-Row Query Result: VB.....	4-8
4-17	Using DataSet Class: Declaring Variables: C# .....	4-10
4-18	Using DataSet Class: Declaring Variables: VB.....	4-10
4-19	Disabling the Save Button: C#.....	4-11
4-20	Disabling the Save Button: VB .....	4-11
4-21	Binding Data to the Grid: C#.....	4-11
4-22	Binding Data to the Grid: VB .....	4-11
4-23	Updating DataSet: C#.....	4-12
4-24	Updating DataSet: VB .....	4-12
5-1	Generated SQL Form of the New Table .....	5-5
5-2	Creating a Table Index in SQL .....	5-6
5-3	Adding Foreign Key and Primary Key Constraints to a Table .....	5-8
5-4	Filling Data into the Form: C#.....	5-12
5-5	Filling Data into the Form: VB .....	5-12
5-6	The save_Click() Method: C# .....	5-13
5-7	The save_Click() Method: VB.....	5-14
6-1	PL/SQL Code for Package HR_DATA.....	6-4
6-2	Assigning Reference Cursors .....	6-5
6-3	Changing OracleCommand to Use a Stored Procedure: C#.....	6-8
6-4	Changing OracleCommand to Use a Stored Procedure: VB .....	6-8
6-5	Defining and Binding OracleParameter Objects for Stored Procedure: C#.....	6-8
6-6	Defining and Binding OracleParameter Objects for Stored Procedure: VB.....	6-9
7-1	Adding getDepartmentno() Method Code: C#.....	7-5

7-2	Adding getDepartmentno() Method Code: VB .....	7-6
8-1	Setting NLS_TERRITORY and NLS_LANGUAGE Parameters: United States .....	8-2
8-2	Testing the NLS Date Format Settings .....	8-2
8-3	Setting NLS_TERRITORY and NLS_LANGUAGE Parameters: Germany .....	8-3
8-4	Setting NLS_TERRITORY Parameter: United States .....	8-3
8-5	Testing NLS Number Format Settings .....	8-3
8-6	Setting NLS_TERRITORY Parameter: Germany .....	8-4
8-7	Setting NLS_SORT Parameter: Binary .....	8-4
8-8	Testing NLS Sort Order Settings .....	8-4
8-9	Setting NLS_SORT Parameter: Spanish .....	8-5
8-10	Setting NLS_LANGUAGE Parameter: United States .....	8-5
8-11	Testing NLS Error Messages Settings .....	8-6
8-12	Setting NLS_LANGUAGE Parameter: French .....	8-6
8-13	How to Obtain Oracle Globalization Settings: C# .....	8-7
8-14	How to Obtain Oracle Globalization Settings: VB .....	8-8
8-15	Disabling the Change Button: C# .....	8-9
8-16	Disabling the Change Button: VB .....	8-9
8-17	Creating an OracleGlobalization Object: C# .....	8-10
8-18	Creating an OracleGlobalization Object: VB .....	8-10
8-19	Retrieving the Globalization Session Information: C# .....	8-10
8-20	Retrieving the Globalization Session Information: VB .....	8-10
8-21	Changing the Date Format and Updating the DataSet: C# .....	8-11
8-22	Changing the Date Format and Updating the DataSet: VB .....	8-11

---

# Preface

## Audience

This document is intended as an introduction to application development with Oracle Database Express Edition in Microsoft .NET. We assume that users of this book have already read the *Oracle Database Express Edition 2 Day DBA* and the *Oracle Database Express Edition 2 Day Developer Guide*, are familiar with basics of SQL and PL/SQL, and know how to use Microsoft Visual Studio .NET.

## Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility/>

### Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

### Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

## Related Documents

For more information, see the following documents in Oracle Database XE documentation set:

- *Oracle Data Provider for .NET Developer's Guide*
- *Oracle Database Extensions for .NET Developer's Guide*
- *Oracle Database Express Edition 2 Day DBA*
- *Oracle Database Express Edition 2 Day Developer Guide*
- Dynamic help, which is part of the Oracle Developer Tools for Visual Studio .NET

## Conventions

The following text conventions are used in this document:

Convention	Meaning
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.



---

# Introduction

This chapter provides a brief description of Oracle technologies for the Microsoft .NET Framework.

This chapter contains the following sections:

- [What is Microsoft .NET Framework](#)
- [Introduction to Oracle Data Provider for .NET](#)
- [Introduction to Oracle Developer Tools for Visual Studio .NET](#)
- [Introduction to .NET Stored Procedures](#)

This book is conceived as a quick start guide, to describe the key features of Oracle Data Provider for .NET and Oracle Developer Tools for Visual Studio .NET. It leads you through installation and configuration, shows how to build basic applications using Oracle Data Provider for .NET and Oracle Developer Tools for Visual Studio .NET, and how to create and use both PL/SQL and .NET stored procedures.

After working through this book, you will be ready to continue with more extensive information available in the Oracle Database documentation library.

#### See Also:

- Dynamic help available within Visual Studio .NET
- *Oracle Data Provider for .NET Developer's Guide*
- *Oracle Database Extensions for .NET Developer's Guide*
- *Oracle Database Express Edition 2 Day DBA*
- *Oracle Database Express Edition 2 Day Developer Guide*

## What is Microsoft .NET Framework

The Microsoft .NET Framework, .NET, is a multi-language environment for building, deploying, and running XML Web services and applications. Its main components are:

#### Common Language Runtime

Common Language Runtime, or CLR, is a language-neutral development and run-time environment that provides services that help manage running applications

#### Framework Class Libraries

The Framework Class Libraries, or FCL, provide a consistent, object-oriented library of prepackaged functionality.

## Introduction to Oracle Data Provider for .NET

Oracle Data Provider for .NET (ODP.NET) provides data access from .NET client applications to Oracle databases. ODP.NET data access is fast, efficient, and includes access to other Oracle Database features.

## Introduction to Oracle Developer Tools for Visual Studio .NET

Oracle Developer Tools for Visual Studio .NET is a set of application tools that are integrated with the Visual Studio .NET environment. These tools provide graphical user interface access to Oracle functionality, enable the user to perform a wide range of application development tasks, and improve development productivity and ease of use. Oracle Developer Tools support the programming and implementation of .NET stored procedures using Visual Basic, C#, and other .NET languages.

Oracle Developer Tools include the Oracle Explorer for browsing the Oracle schema, designers and wizards to create and alter schema objects, and the ability to drag and drop schema objects onto a .NET form to automatically generate code. Additional features include a PL/SQL editor with integrated context-sensitive dynamic help, and an Oracle Data Window for performing routine database tasks like inserting and updating data or testing stored procedures in the Visual Studio environment. For maximum flexibility, there is also an Oracle Query Window for executing SQL statements or PL/SQL scripts.

## Introduction to .NET Stored Procedures

Oracle Database Extensions for .NET has the following features:

### **Common Language Runtime Host for Oracle Database**

The Oracle Database on Windows hosts the Microsoft Common Language Runtime (CLR). The integration of Oracle Database with CLR enables applications to run .NET stored procedures or functions on Oracle Database with Microsoft Windows Server 2003, Windows 2000, and Windows XP.

### **Data Access for .NET Classes through Oracle Data Provider for .NET**

Using Microsoft Visual Studio .NET, you can build .NET procedures or functions into a .NET assembly. All .NET stored procedures and functions use ODP.NET to access data.

### **Oracle Deployment Wizard for Visual Studio .NET**

After building .NET procedures and functions into a .NET assembly, deploy them in Oracle Database using the Oracle Deployment Wizard for .NET, a component of the Oracle Developer Tools for Visual Studio .NET.

---

# Installing Oracle Developer Tools

This chapter demonstrates the installation of Oracle Developer Tools.

This chapter contains the following sections:

- [What You Need](#)
- [Installing Oracle Developer Tools](#)
- [Unlocking the User Account](#)
- [Uninstalling Oracle Developer Tools](#)

## What You Need

### Sample Data

The sample data used in this book ships with Oracle Database XE, and installs out of the box. The sample data that ships with the product is the HR component of the Sample Schemas.

**See Also:** *Oracle Database Sample Schemas* for the HR data model and table descriptions

### Oracle Database XE Server

You should install a copy of Oracle Database XE server on your computer. Oracle Database XE server is a free Oracle database that is available for download from the Oracle Database XE Web site, at:

<http://www.oracle.com/technology/x>

It has a browser-based user interface, Oracle Application Express, for administering the database, running scripts and queries, building Web-based applications, and more.

Note that the installation of Oracle Database XE includes Oracle Data Provider for .NET, .NET Stored Procedures, OLE DB and ODBC. It does not include Oracle Developer Tools, a set of application tools integrated with the Visual Studio .NET development environment, which enable you to perform a wide range of application development tasks.

### Oracle Database XE Client

The Oracle Database XE client is installed as part of your Oracle Database XE server. The Oracle Database XE client can also be installed by itself to access a remote server. The client includes all the data access drivers that ship with Oracle Database XE, such as Oracle Data Provider for .NET.

### Visual Studio .NET 2003

Before proceeding with instructions in this book, you should have a complete installation of the Visual Studio .NET 2003. If you wish to deploy an existing .NET application, you only need the .NET Framework.

## Installing Oracle Developer Tools

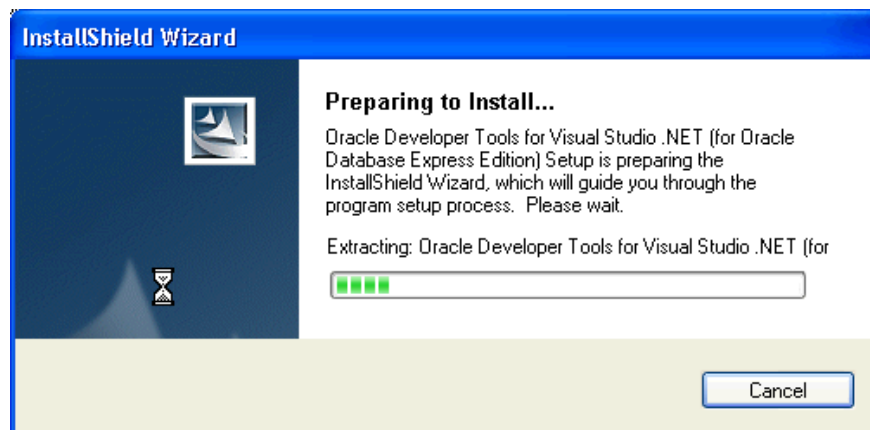
Before proceeding with these steps to install Oracle Developer Tools for Visual Studio .NET, you must have a full installation of Oracle Database XE or the Oracle XE client alone, and Visual Studio .NET:

1. In your Internet browser, navigate to the following software download location:

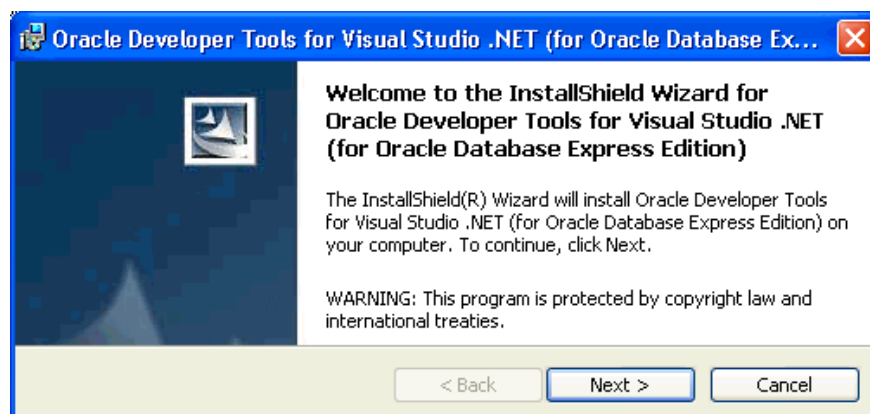
[http://www.oracle.com/technology/software/tech/dotnet/odt\\_xe\\_index.html](http://www.oracle.com/technology/software/tech/dotnet/odt_xe_index.html)

2. Download the `setup.exe` file to your desktop.
3. Double-click the **Setup** icon.

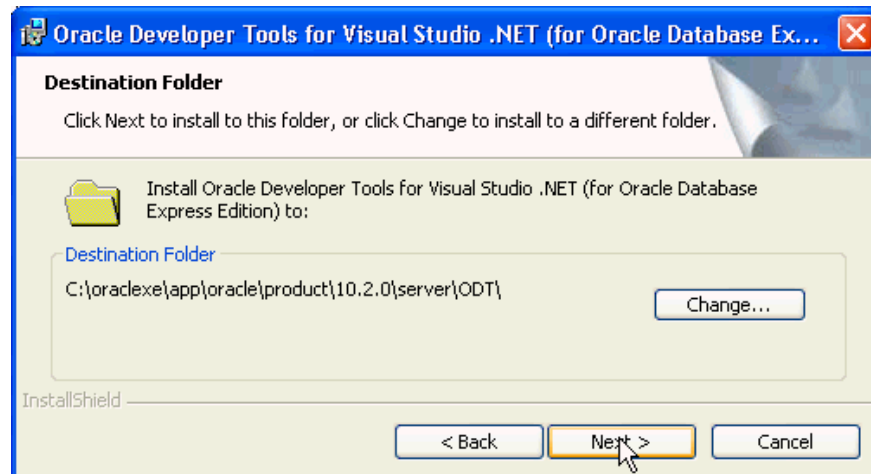
The InstallShield Wizard will be launched.



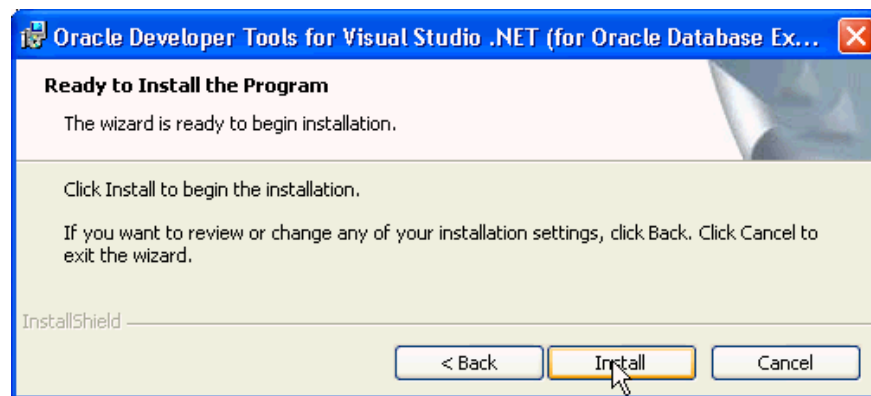
4. Once the Oracle Developer Tools for Visual Studio .NET InstallShield Wizard window appears, click **Next**.



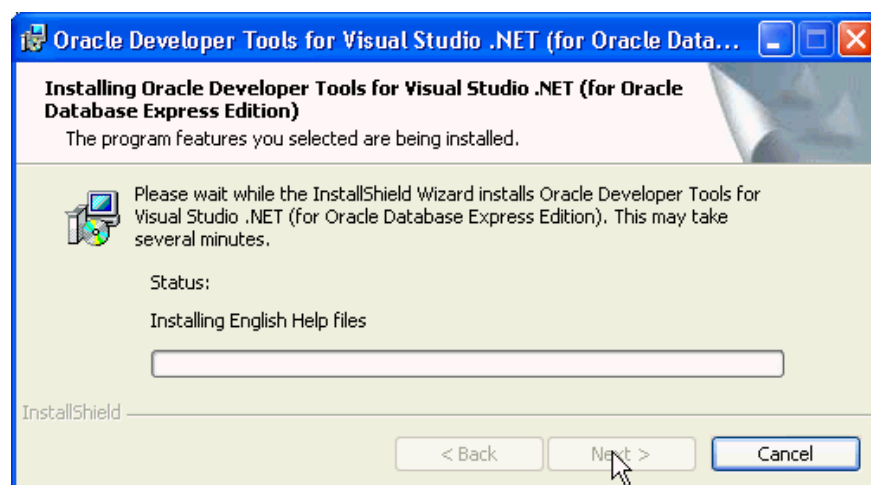
5. On the Destination Folder window, accept the default installation location and click **Next**.



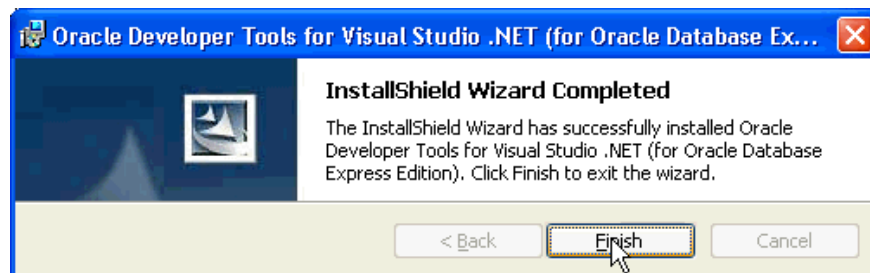
6. On the Ready to Install the Program window, click **Install**.



7. The Installing Oracle Developer Tools for Visual Studio .NET (for Oracle Database Express Edition) window appears.



8. Click **Finish** to complete the installation.



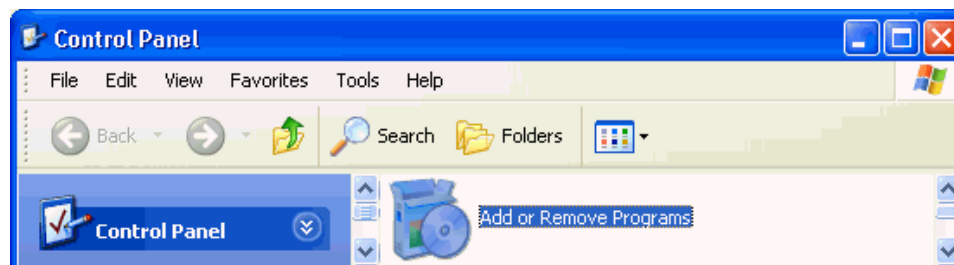
## Unlocking the User Account

The applications in this book connect to the Oracle Database XE as the hr user. You may need to unlock the hr account while signed in as a user with database administrator privileges. To use the Oracle Application Express, see Chapter 6, "Managing Users and Security" in the *Oracle Database Express Edition 2 Day DBA*.

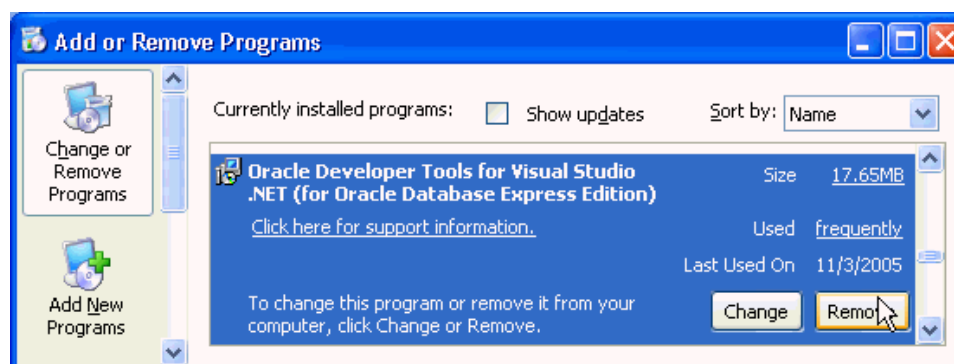
## Uninstalling Oracle Developer Tools

If you need to remove Oracle Developer Tools from your computer, follow these steps:

1. Open the Windows Control Panel.
2. Double-click the **Add or Remove Programs** icon.



3. Select **Oracle Developer Tools for Visual Studio .NET** from the list of programs, and click the **Remove** button.



4. On the Add or Remove Programs confirmation dialog box, click **Yes**.

---

## Connecting to the Database

This chapter explains how to connect to the Oracle database. You will be using the application you will build in this chapter as a starting point for work in all subsequent chapters.

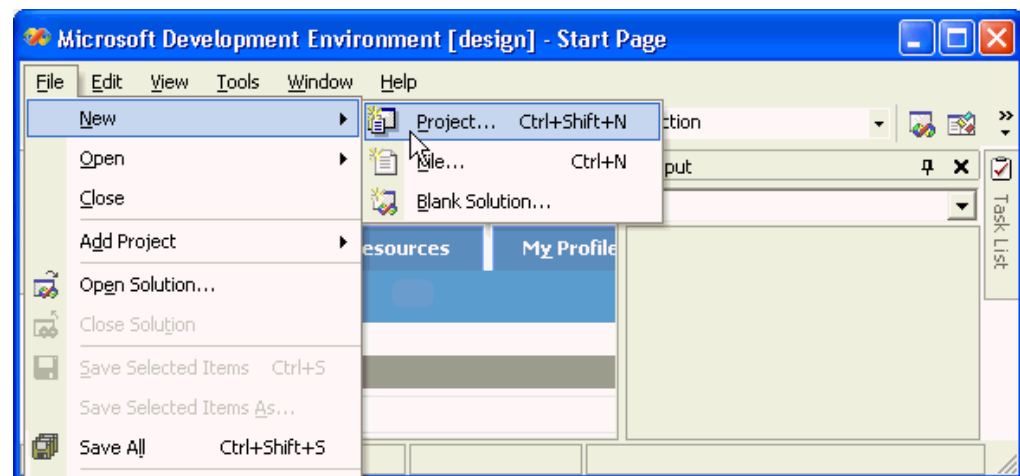
This chapter contains the following sections:

- [Starting a New Project](#)
- [Adding a Reference](#)
- [Adding Initial Programmatic Statements](#)
- [Adding Connection Elements](#)
- [Using the Connection Object](#)
- [Running the Application](#)
- [Error Handling](#)
- [Closing the Database Connection](#)

### Starting a New Project

Follow these steps to start a project in Visual Studio .NET 2003:

1. Click the **New Project** button. Alternatively, from the **File** menu, select **New**, and then select **Project**.



A **New Project** dialog box appears.

2. On the left side of the **New Project** dialog box under **Project Types**, select **Visual C# Projects**.

On the right side of the New Project dialog box under **Templates**, select **Windows Application**.

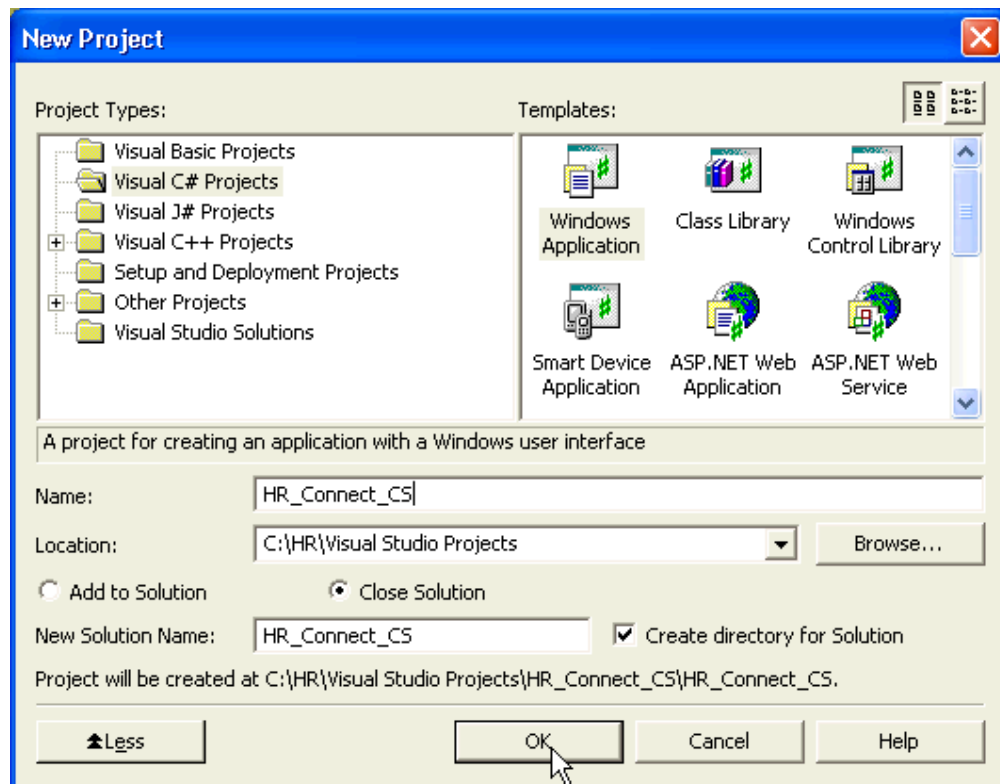
For **Name**, enter `HR_Connect_CS`.

For **Location**, enter `C:\HR\Visual Studio Projects`.

Check the **Create Directory for Solution** box.

For **New Solution Name**, enter `HRApplication`. A solution can contain several projects; when it contains only one project, you can use the same name for both.

Click **OK**.



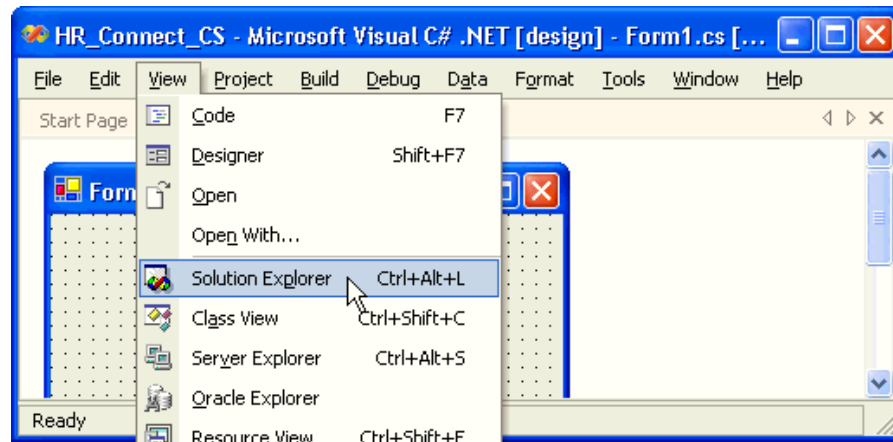
If you wish to create project in Visual Basic, select **Visual Basic Projects** in **Project Types** instead, and enter `HR_Connect_VB` under **Name**.

## Adding a Reference

To connect the project to an Oracle database, you must add a reference to the `Oracle.DataAccess.dll`, which contains the data provider.

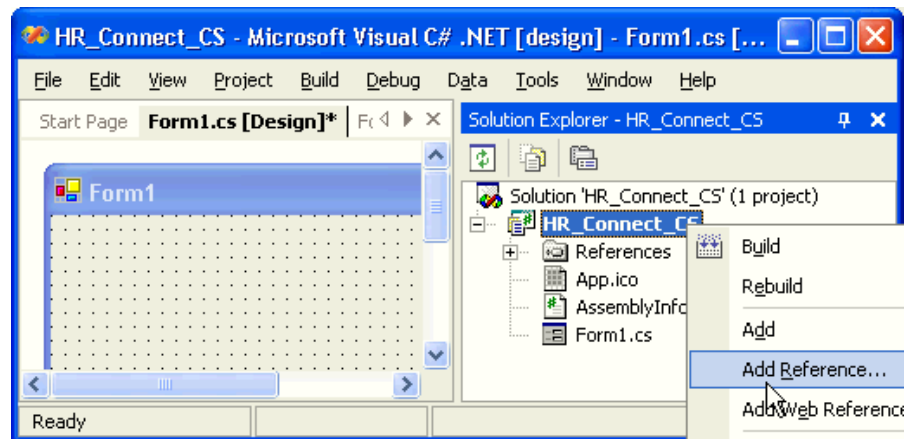
1. If it is not already active, start the Solution Explorer; from the **View** menu, select **Solution Explorer**.





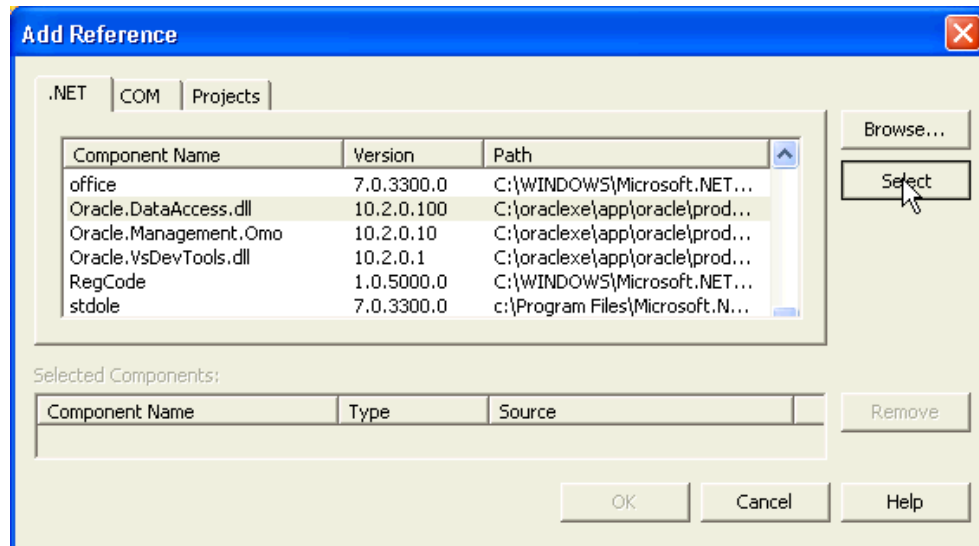
The Solution Explorer appears in the window.

2. In the **Solution Explorer**, select the **References** node, right click and select **Add Reference**. Alternatively, select **Add Reference** from the **Project** menu.



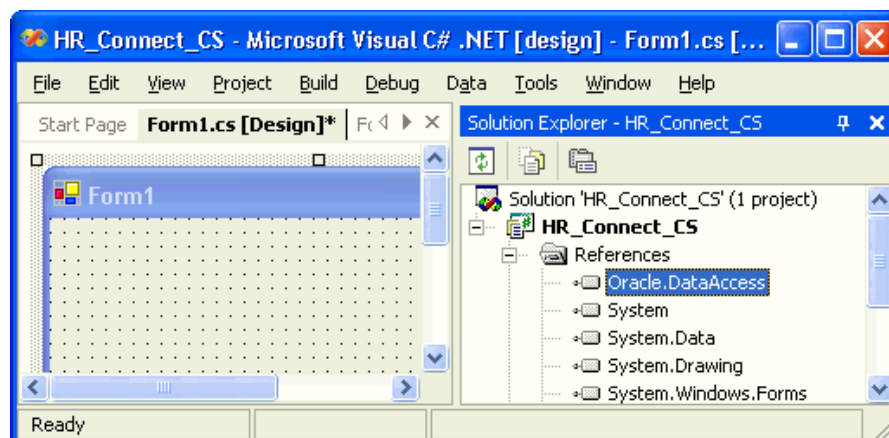
The Add Reference dialog box appears.

3. Scroll down the list of references (under Component Name), and select `Oracle.DataAccess.dll`.  
Click the **Select** button.



Click the **OK** button to add the Oracle Data Provider for .NET to your project.

Note that the **Solution Explorer** window now shows `Oracle.DataAccess` in the **References** folder.

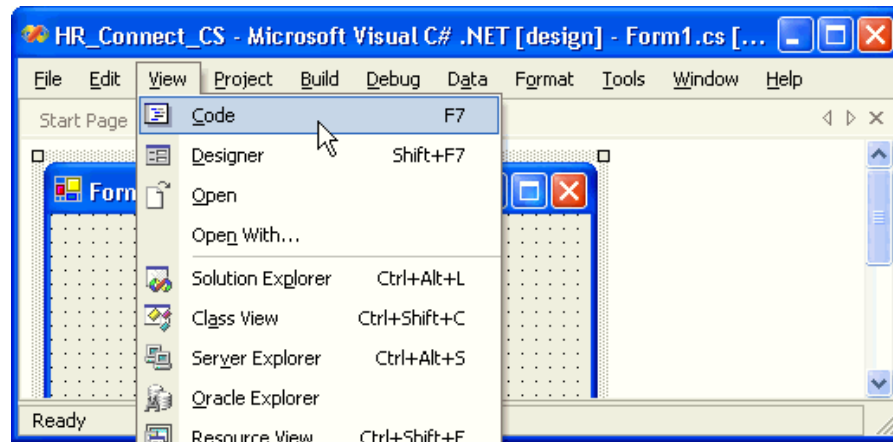


Close the Solution Explorer.

## Adding Initial Programmatic Statements

C# `using` statements and Visual Basic `Imports` statements allow you to refer to database objects without using lengthy, fully qualified names. By convention, these statements appear at or near the top of a code file, before the namespace or class declaration.

1. With `Form1` active, in **View** menu select **Code**, or use the **F7** keyboard shortcut.



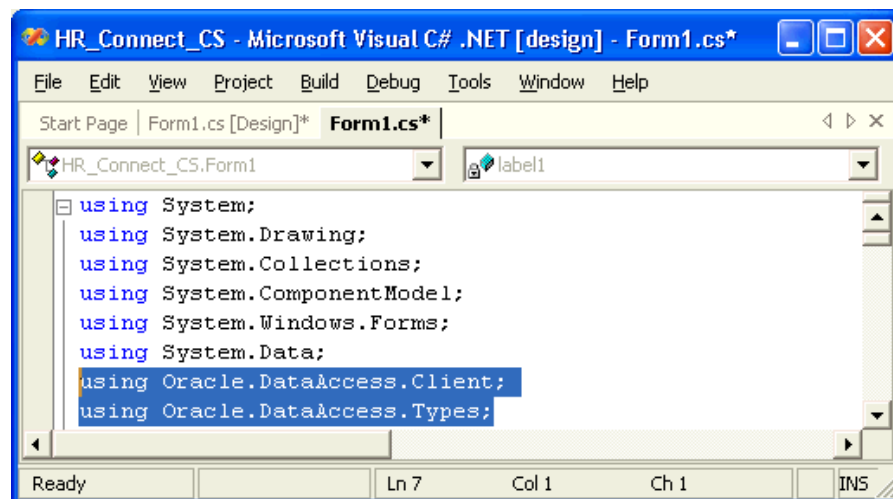
2. Add the statements in [Example 3-1](#) or [Example 3-2](#) to the top of the file.

**Example 3-1 Adding Initial Programmatic Statements: C#**

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

**Example 3-2 Adding Initial Programmatic Statements: VB**

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

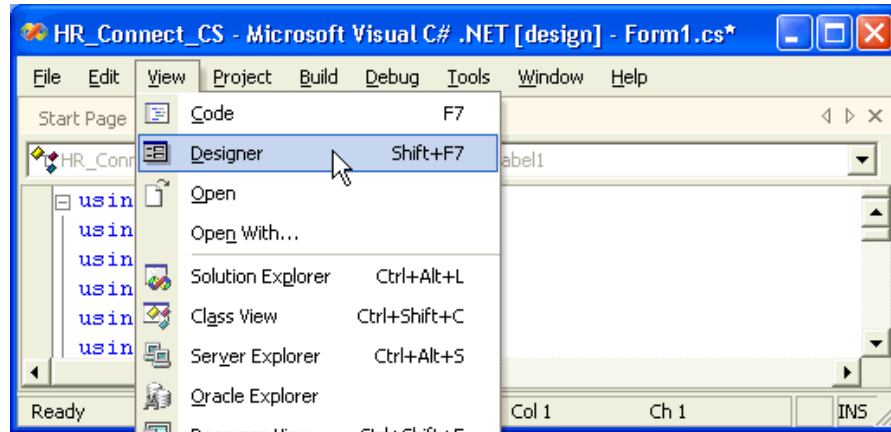


Save the changes using the **Save** icon near the top of the window. Alternatively, from the **File** menu, select **Save**, or use the **Ctrl+S** keyboard shortcut.

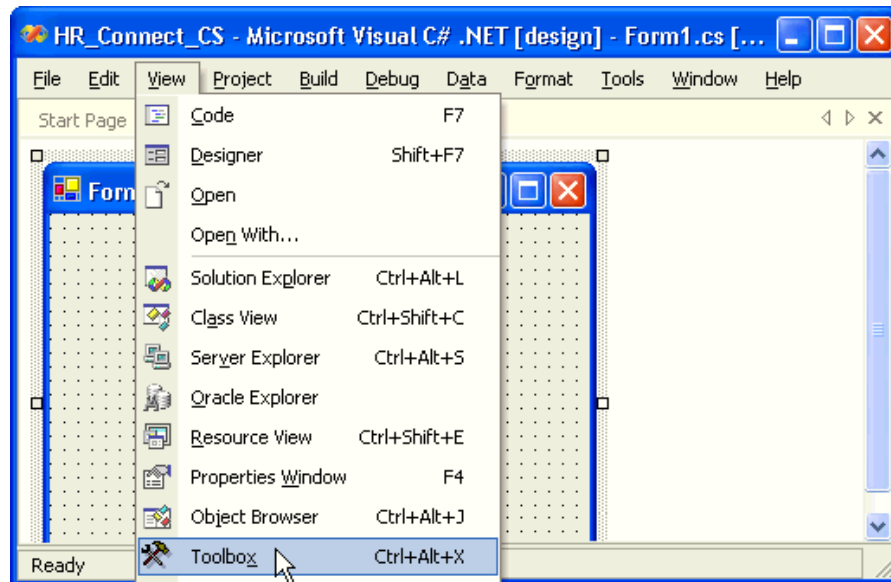
## Adding Connection Elements

To create a connection interface, you must add the necessary data entry elements to the design form.

1. With **Form1** active, change to design view: from the **View** menu, select **Designer**. Alternatively, use the **Shift+F7** keyboard shortcut. You may also wish to close the **Solution Explorer** at this time.

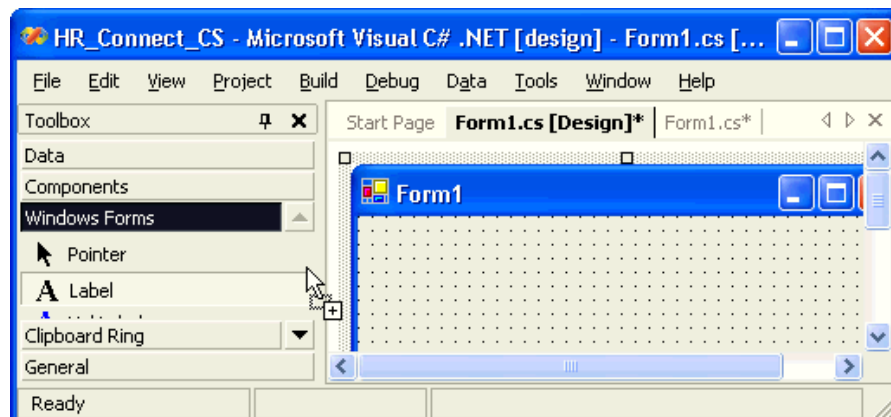


2. From the **View** menu, select **Toolbox**.



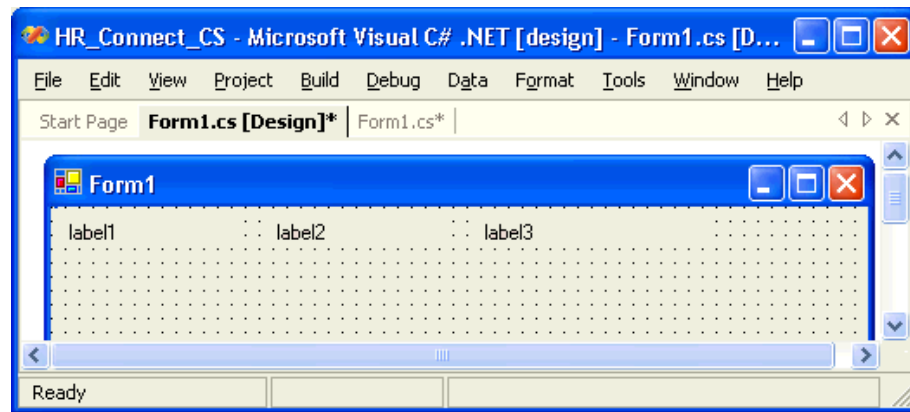
A Toolbox window appears.

3. From the Toolbox, under Windows Forms, select a **Label** and drag it onto **Form1**.

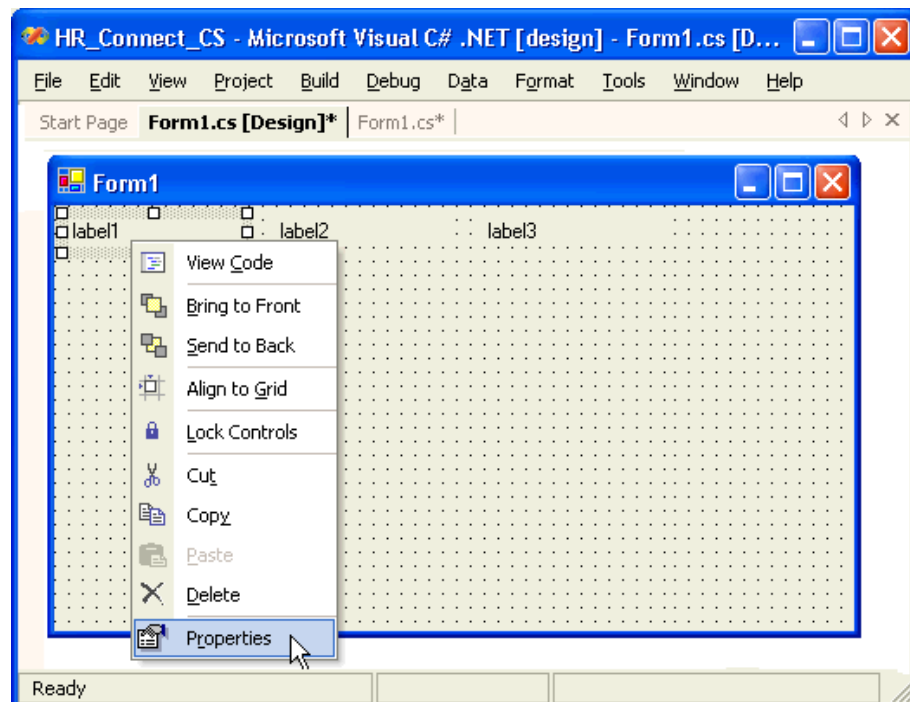


A new label, **label1**, appears on the form.

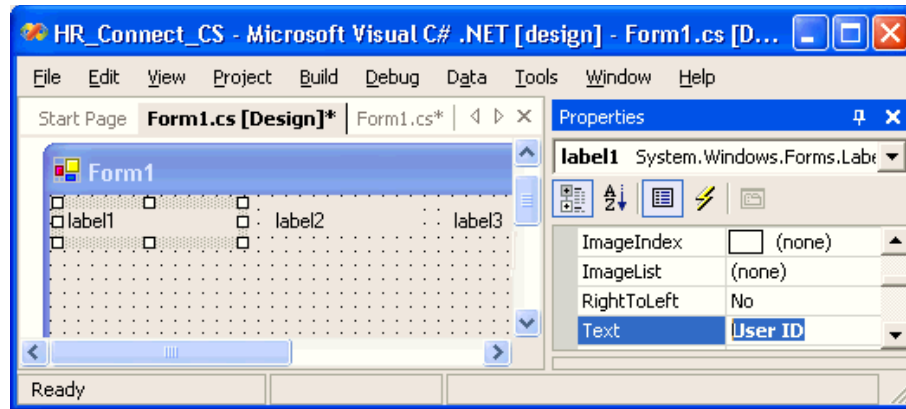
4. Repeat Step 3 twice, adding two more labels to the form (**label2** and **label3**).  
Close the **Toolbox**.



5. Right-click **label1**, and select **Properties**.

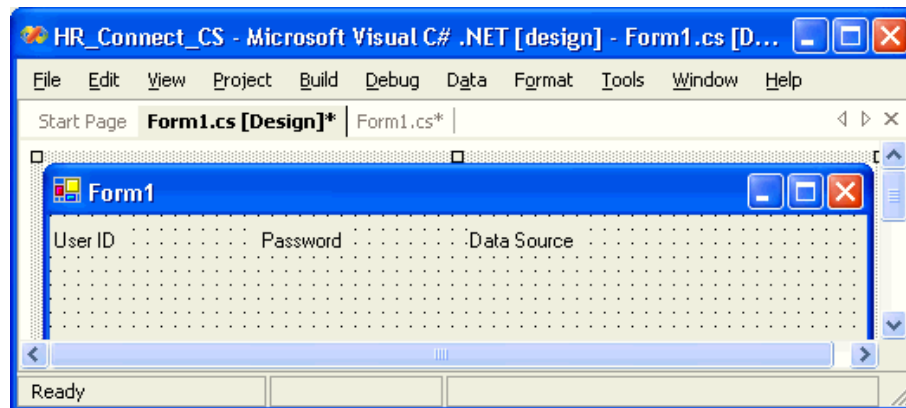


6. Change the Text property from **label1** to **User ID**.

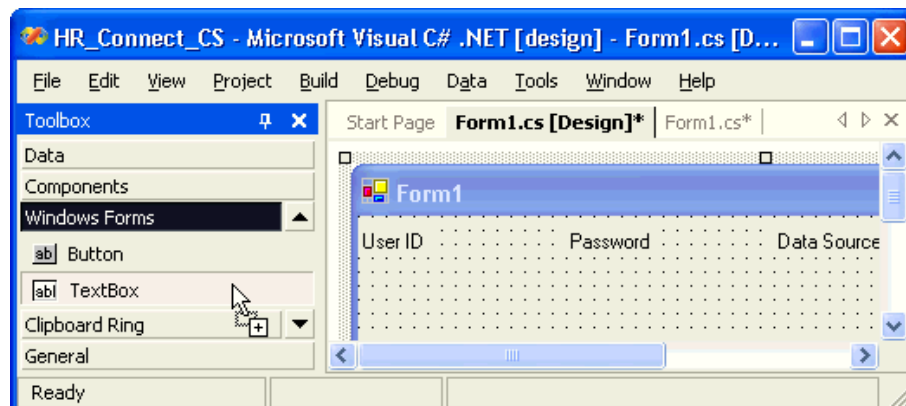


7. Repeat Steps 5 and 6 for the other two labels, changing the text to `Password` and `Data Source`, respectively.

Close the Properties window.



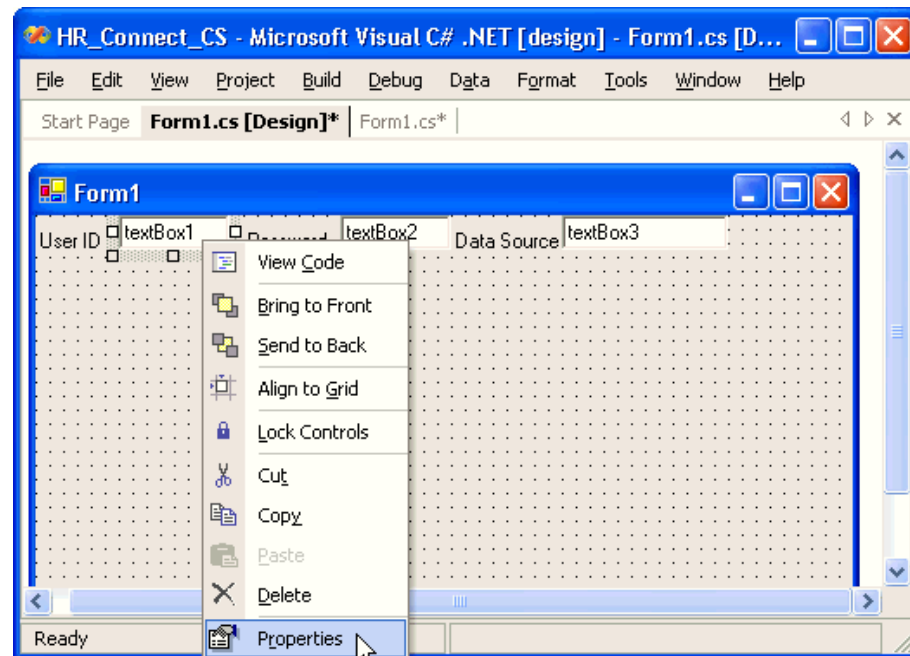
8. From the **View** menu, select **Toolbox**. From the Toolbox, under Windows Forms, select a **Text Box** and drag it onto Form1.



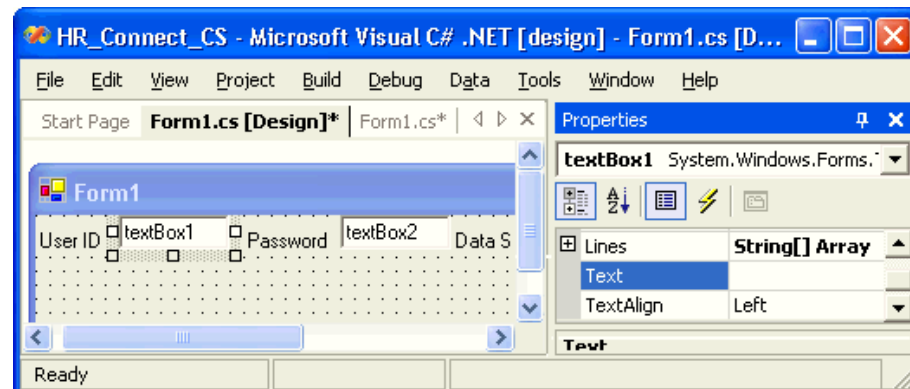
A new text box, `textBox1`, appears on the form.

9. Repeat Step 8 twice, adding two more text boxes (`textBox2` and `textBox3`).  
Close the Toolbox.

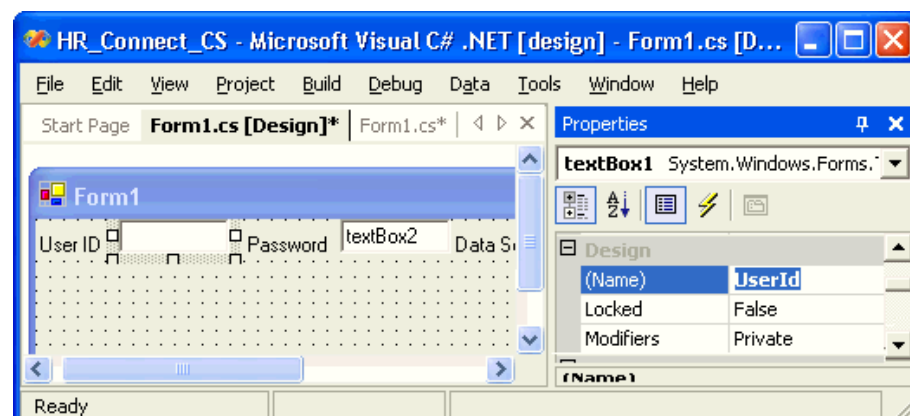
10. Right-click **textBox1**, and select **Properties**.



11. In Properties, under Appearance, remove the text in the **Text** property.

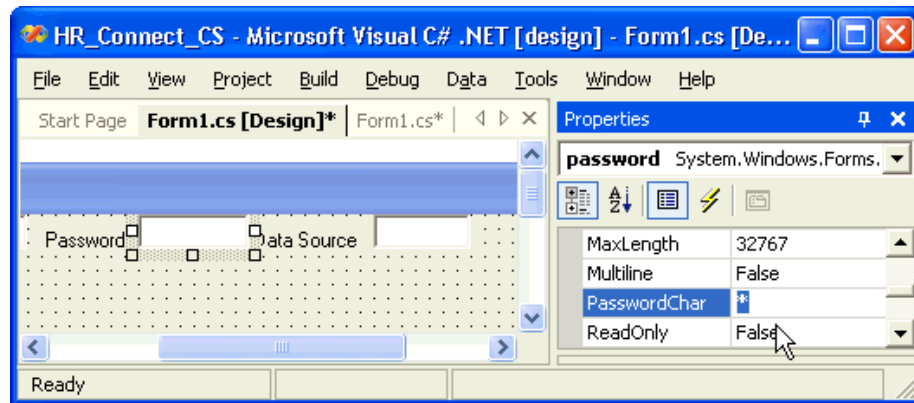


12. In Properties, under Design, change the value of **(Name)** to **userID**.

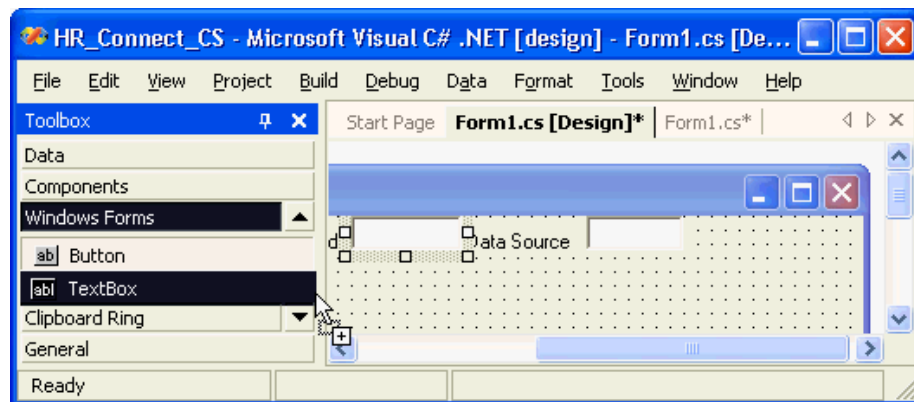


13. Repeat Steps 10 through 12 twice, changing `textBox2` into `password` and `textBox2` into `dataSource`.

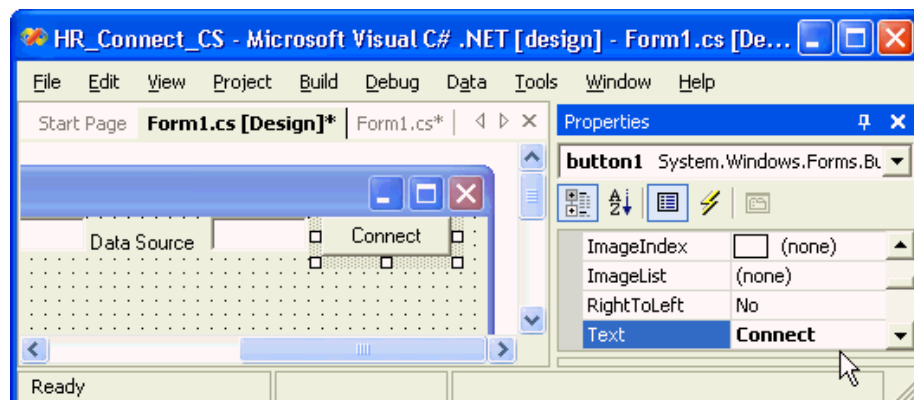
For the password text box, under **Behavior** category, change the value of **PasswordChar** property to `*`. This will hide the password when it is entered.



14. Close the **Properties** window.
15. From the **View** menu, select **Toolbox**. In the Toolbox, under Window Forms, select **Button** and drag it onto Form1.

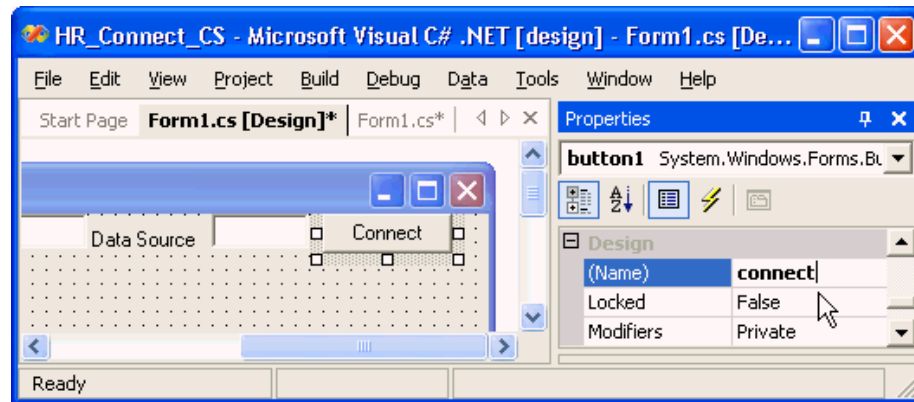


16. Right-click **button1**, and select **Properties**. The Properties window appears.
17. In the Properties window, change the **Text** property to `Connect`.

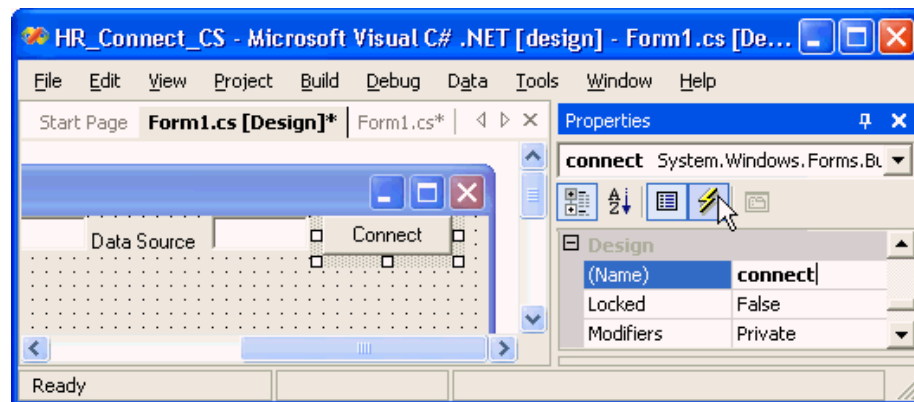




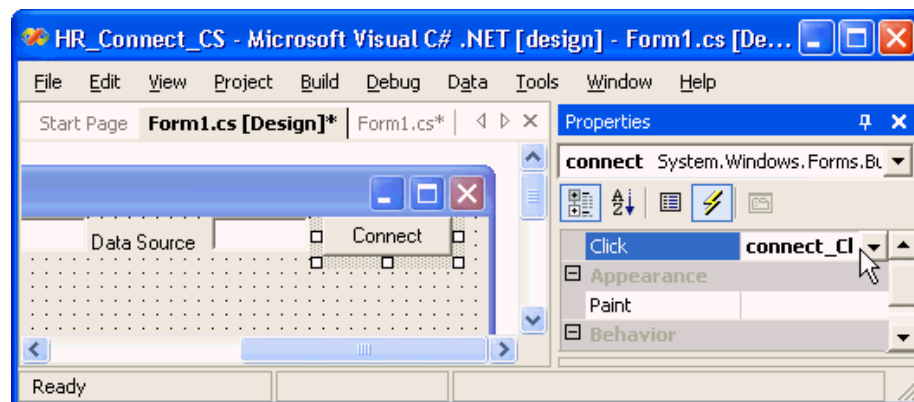
18. Under **Design**, change the **(Name)** property to connect.



19. Save the changes using the **Ctrl+S** keyboard shortcut.
20. Click the lightning icon (**Events**) at the top of the **Properties** window.



21. Ensure that the **Click** event is called `connect_Click`. Save changes.



## Using the Connection Object

The `OracleConnection` object specifies the Oracle Database used by the application.

1. The Easy Connect naming method enables application clients to connect to a database without using any configuration files, simply by specifying the data source attribute through syntax shown in [Example 3-3](#):

**Example 3-3 Easy Connect Naming Method Syntax for Data Source**

```
user id=id;password=psswd;data source=host:port/service_name
```

Where:

- *id* is the user id; we will use `hr` to access the HR schema.
- *psswd* is the password; in this book, we use `hr` password for the HR schema.
- *host* is the DNS name of the server machine to which the XE client will make the connection, such as `hr-server` in the following example of valid connections.
- *port* [optional], if not specified, uses the default value of 1521, which is the server port number from which the client connects to the database.
- *service\_name* [optional], if not specified, the EZ Connect Adapter for XE client will connect to the default service on the host, preconfigured as `XE` in the `listener.ora` file on the XE server.

Note that the default service is a new feature for Oracle Database XE. If you used other Oracle client software, such as Instant Client for Oracle Database Enterprise Edition, you must supply the service name.

Some valid connection strings include:

```
user id=hr;password=hr;data source=hr-server
user id=hr;password=hr;data source=hr-server:1521
user id=hr;password=hr;data source=hr-server:1521/XE
```

2. [Example 3-4](#) and [Example 3-5](#) show how to instantiate a database connection string.

**Example 3-4 Creating an OracleConnection Object: C#**

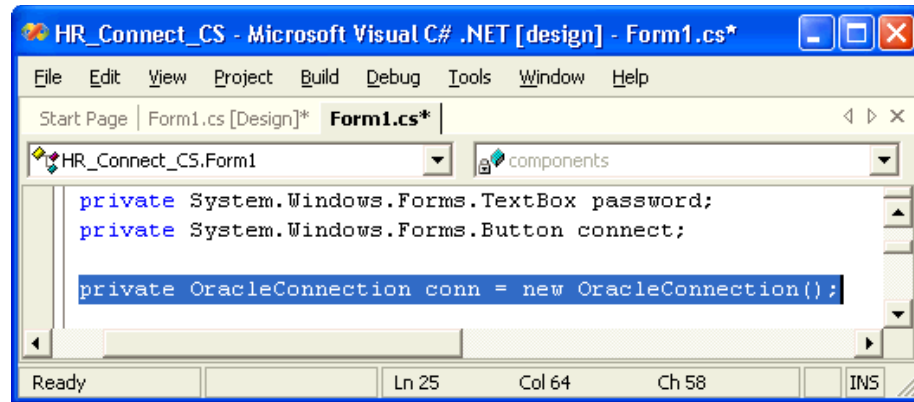
Add this class variable after the GUI elements you created in the previous section.

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

**Example 3-5 Creating an OracleConnection Object: VB**

Add this class variable after the line that starts as `Inherits`, near the top of the file, in the `Form1` class declaration.

```
Dim conn As New OracleConnection
```



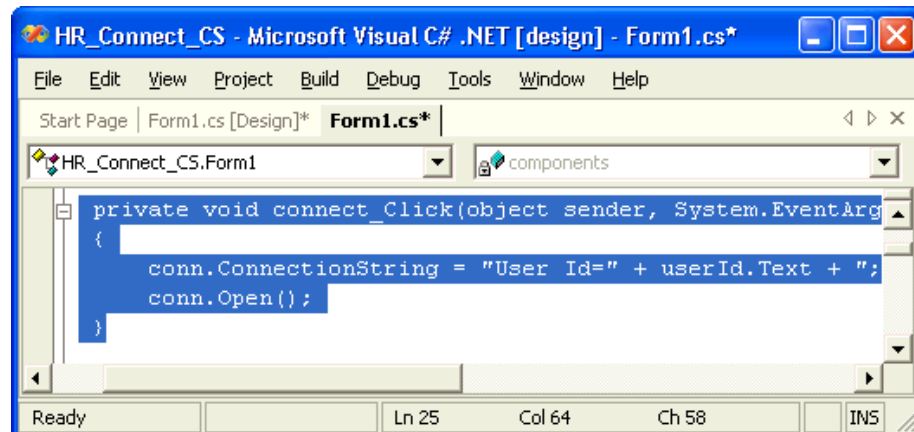
3. Before a connection can be open, it must be built from user input for the User ID, Password, and Data Source. Add the code in [Example 3-6](#) or [Example 3-7](#) to your application. Note that the `Open()` method makes the actual connection. Add this code into the `connect_Click()` method (for C#) or subroutine (VB).

**Example 3-6 Building and Opening a Connection: C#**

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

**Example 3-7 Building and Opening a Connection: VB**

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



4. As part of good programming practice, add the code in [Example 3-8](#) and [Example 3-9](#) after the `Open()` call of **Form1**. This will disable the **Connect** button after a connection is successfully made.

**Example 3-8 Disabling the Connect Button: C#**

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

**Example 3–9 Disabling the Connect Button: VB**

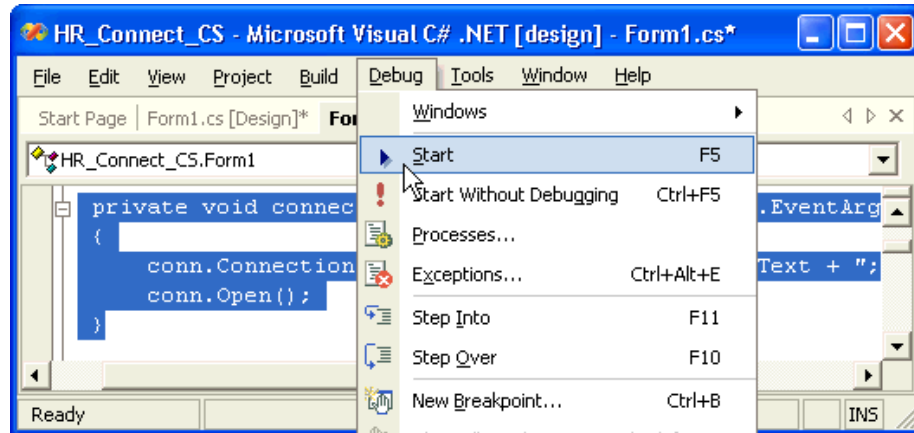
```
connect.Enabled = false
```

5. Save the application.

## Running the Application

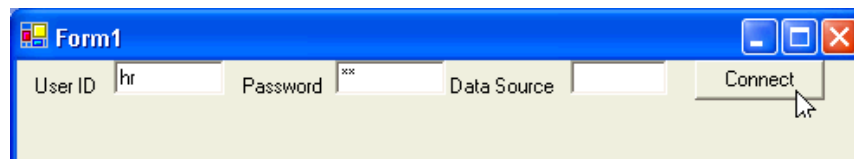
The following steps show how to run the application created in the previous sections.

1. With Form1 active, from the **Debug** menu, select **Start**. Alternatively, use the F5 keyboard shortcut.

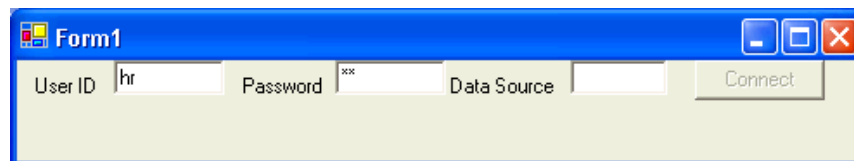


After the application is built successfully, the Form1 window appears.

2. Enter the user ID, the password, and the data source information, and click **Connect**. If you are using a local database, located on the same machine as the .NET client, you may leave the data source field blank.



3. Note that after a successful connection, the **Connect** button is disabled.



## Error Handling

Oracle Data Provider for .NET contains three classes for error handling and support:

- The `OracleError` class represents a warning or an error reported by Oracle.

- An `OracleErrorCollection` class represents a collection of all errors that are thrown by the Oracle Data Provider for .NET. In fact, it is a simple `ArrayList` that holds a list of `OracleErrors`.
  - The `OracleException` class represents an exception that is thrown when the Oracle Data Provider for .NET encounters an error. Each `OracleException` object contains at least one `OracleError` object in the `Error` property that describes the error or warning.
1. The .NET languages use Try-Catch-Finally structured error handling. Change the code in Form1, as indicated in [Example 3-10](#) and [Example 3-11](#), which are simple implementations of the Try-Catch-Finally syntax.

**Example 3-10 Error Handling with Try-Catch-Finally Syntax: C#**

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

catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}

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

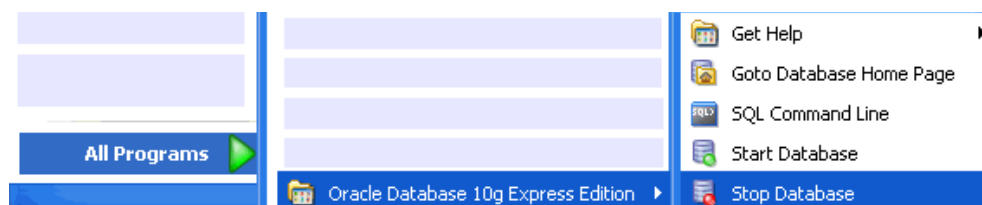
**Example 3-11 Error Handling with Try-Catch-Finally Syntax: VB**

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

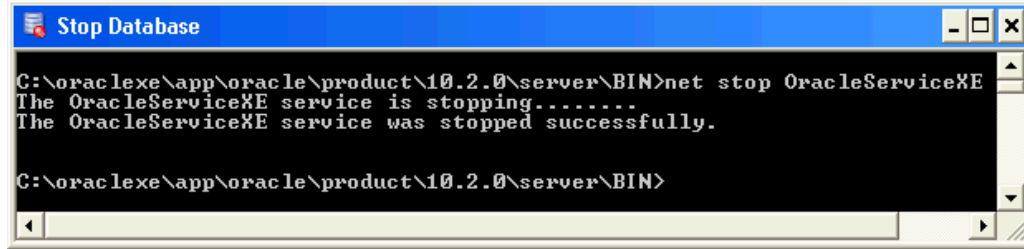
Catch ex As Exception
    MessageBox.Show(ex.Message.ToString())

Finally
    conn.Dispose()
End Try
```

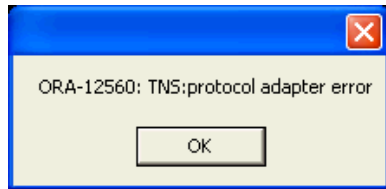
2. Before testing this code, stop the database. From the **Start** button, select **All Programs**, then select **Oracle Database 10g Express Edition**, and select **Stop Database**.



3. The database will begin to shut down. You should see a **Stop Database** window. Do not proceed with the following steps until it indicates that the "OracleServiceXE service was stopped successfully".



4. Run the application again, as described in section "Running the Application" on page 3-14, and attempt to connect. The error caught when the database is unavailable appears as "ORA-12560: TNS:protocol adapter error".



While this approach will capture errors encountered when connecting to the database, the message is not very informative for the end user.

5. Add another catch statement to trap common database errors and to display these errors in a more user-friendly manner. Insert [Example 3-12](#) or [Example 3-13](#) code before the generic catch statement.

**Example 3-12 Catching Common Database Error Messages: C#**

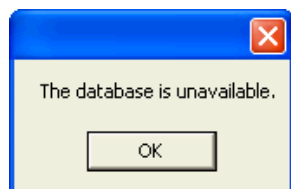
```
catch (OracleException ex)
{
    switch (ex.Number)
    {
        case 1:
            MessageBox.Show("Error attempting to insert duplicate data.");
            break;
        case 12560:
            MessageBox.Show("The database is unavailable.");
            break;
        default:
            MessageBox.Show("Database error: " + ex.Message.ToString());
            break;
    }
}
```

**Example 3-13 Catching Common Database Error Messages: VB**

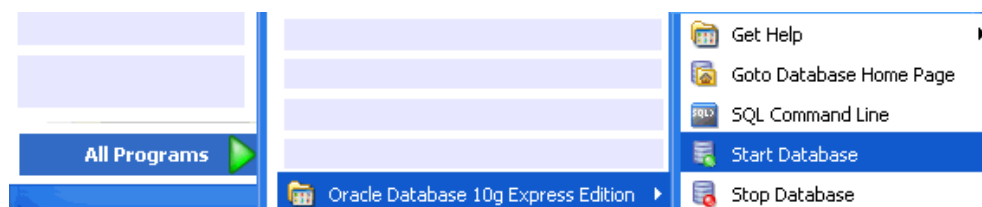
```
Catch ex As OracleException ' catches only Oracle errors
Select Case ex.Number
Case 1
    MessageBox.Show("Error attempting to insert duplicate data.")
Case 12560
    MessageBox.Show("The database is unavailable.")
Case Else
    MessageBox.Show("Database error: " + ex.Message.ToString())
End Select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

The Case statements should be ordered from most specific to most general. If there are no `OracleExceptions`, the first Catch statement branch ([Example 3-12](#) or [Example 3-13](#)) is skipped. The second Catch statement (in [Example 3-10](#) or [Example 3-11](#)) catches all other Exceptions.

6. Run the application again, as described in section "Running the Application" on page 3-14. After implementing [Example 3-12](#) or [Example 3-13](#), the ORA-12560 error appears as "The database is unavailable.":



7. The Finally code block is always executed. If the connection object's `Dispose()` method call is in the Finally code block, the database connection will always be closed after the Try-Catch-Finally block is complete. Attempting to close a closed database connection does not cause an error. If the database is unavailable, the database connection is not opened, so the Finally code block attempts to close a connection that does not exist, making these calls irrelevant. However, placing `Dispose()` in the Finally code block guarantees that the connection is closed.
8. Before proceeding, restart the database. From the **Start** button, select **All Programs**, then select **Oracle Database 10g Express Edition**, and select **Start Database**.



9. The database services will begin to start. You should see a **Start Database** window. Do not proceed with the following steps until it indicates that the "OracleServiceXE service was started successfully".

```

Start Database

C:\oracle\app\oracle\product\10.2.0\server\BIN>net start OracleMTSRecover
ice
The requested service has already been started.
More help is available by typing NET HELPMSG 2182.

C:\oracle\app\oracle\product\10.2.0\server\BIN>net start OracleXETNSList
The requested service has already been started.
More help is available by typing NET HELPMSG 2182.

C:\oracle\app\oracle\product\10.2.0\server\BIN>net start OracleServiceXE
The OracleServiceXE service is starting.....
The OracleServiceXE service was started successfully.

C:\oracle\app\oracle\product\10.2.0\server\BIN>

```

## Closing the Database Connection

1. A connection `Dispose()` method closes and disposes the connection, as shown in [Example 3-14](#) and [Example 3-15](#).

**Example 3-14 Closing and Disposing a Connection: C#**

```
conn.Dispose();
```

**Example 3-15 Closing and Disposing a Connection: VB**

```
conn.Dispose()
```

2. C# has an alternative syntax that disposes of a connection when it goes out of scope, through the `using` keyword, as shown in [Example 3-16](#).

**Example 3-16 Closing and Disposing a Connection when Out of Scope: C#**

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



---

## Building an Oracle Data Provider for .NET Application

This chapter explains how to use Oracle Data Provider for .NET.

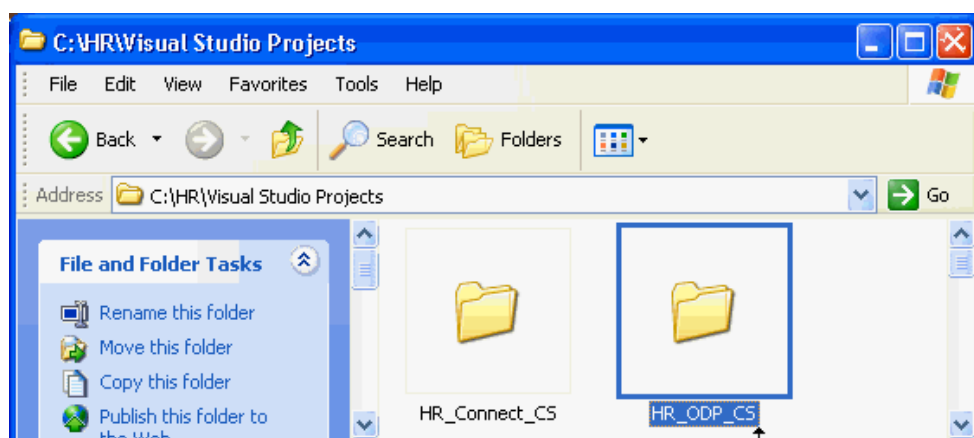
This chapter contains the following sections:

- [Copying a Project](#)
- [Using the Command Object](#)
- [Retrieving Data: a Simple Query](#)
- [Retrieving Data: Bind Variables](#)
- [Retrieving Data: Multiple Values](#)
- [Using the DataSet Class with Oracle Data Provider for .NET](#)
- [Inserting, Deleting and Updating Data](#)

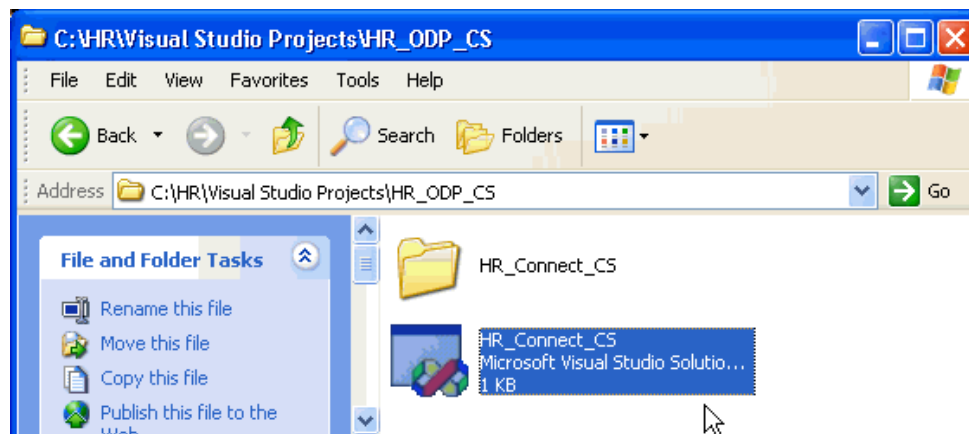
### Copying a Project

For this chapter, you need to use the application developed in [Chapter 3, "Connecting to the Database"](#). Follow these steps to copy the project to a new directory.

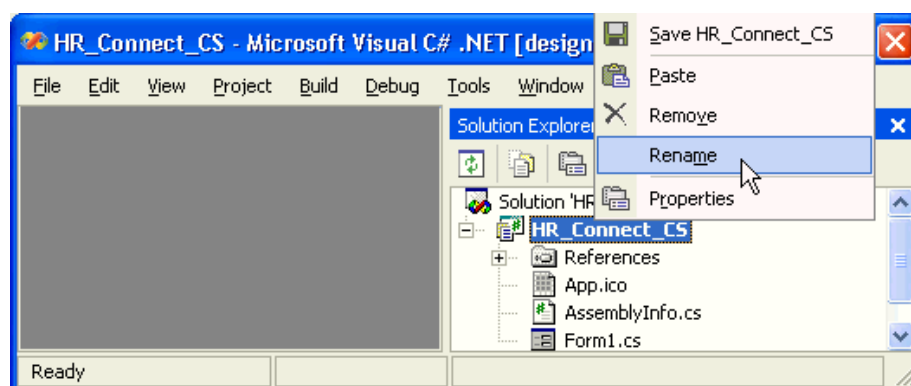
1. Complete all steps in [Chapter 3, "Connecting to the Database"](#).
2. Using the Windows Explorer, navigate to the directory C:\HR\Visual Studio Projects. Make a copy of the entire folder HR\_Connect\_CS (HR\_Connect\_VB for Visual Basic), and rename the new folder HR\_ODP\_CS (HR\_ODP\_VB for Visual Basic).



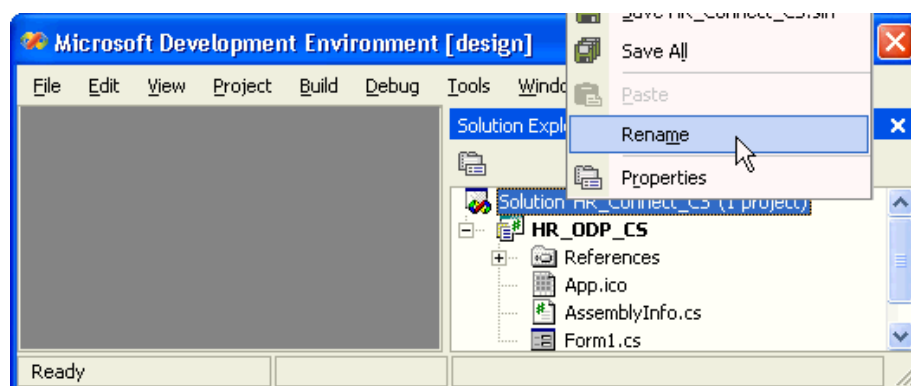
- Open the folder HR\_ODP\_CS. Launch the solution **HR\_Connect\_CS** by double clicking on that icon.



- In the Solution Explorer, right-click **HR\_Connect\_CS**(**HR\_Connect\_VB** for Visual Basic), and select **Rename**. Change the name to HR\_ODP\_CS(**HR\_ODP\_VB** for Visual Basic).



- In the Solution Explorer, right-click **Solution 'HR\_Connect\_CS'** (**Solution 'HR\_Connect\_VB'** for Visual Basic), and select **Rename**. Change the name to HR\_ODP\_CS (**HR\_ODP\_VB** for Visual Basic).



- Close the window. When prompted whether you want to save changes, click **Yes**.
- Launch the **HR\_ODP\_CS**(or **HR\_ODP\_VB**) solution.

## Using the Command Object

The `OracleCommand` class specifies a SQL command, stored procedure, or table name. It creates a database request, sends the request to the database, and returns the result.

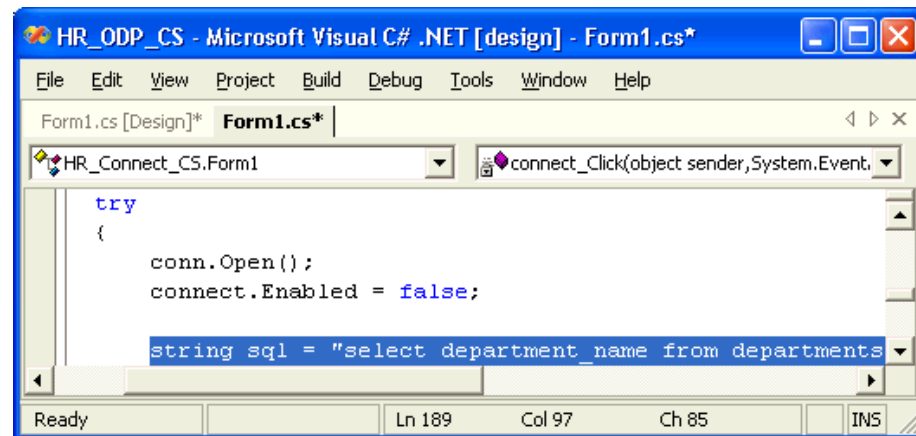
1. Create a string that represents the SQL query, as shown in [Example 4-1](#) and [Example 4-2](#). Add this code to the body of the `try` statement.

### Example 4-1 Creating a SQL Statement String: C#

```
string sql = "select department_name from departments where department_id = 10";
```

### Example 4-2 Creating a SQL Statement String: VB

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



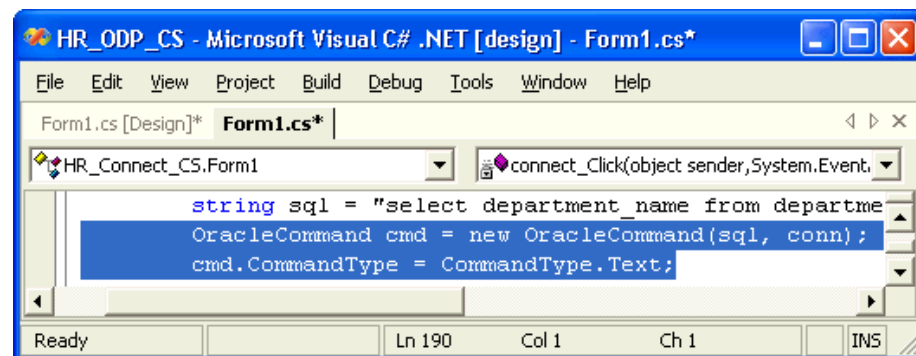
2. Use the new `sql` variable to create the `OracleCommand` object, and set the its `CommandType` property to run a text command.

### Example 4-3 Using a Command to Query the Database: C#

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

### Example 4-4 Using a Command to Query the Database: VB

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



## Retrieving Data: a Simple Query

To retrieve data from the database, follow these steps:

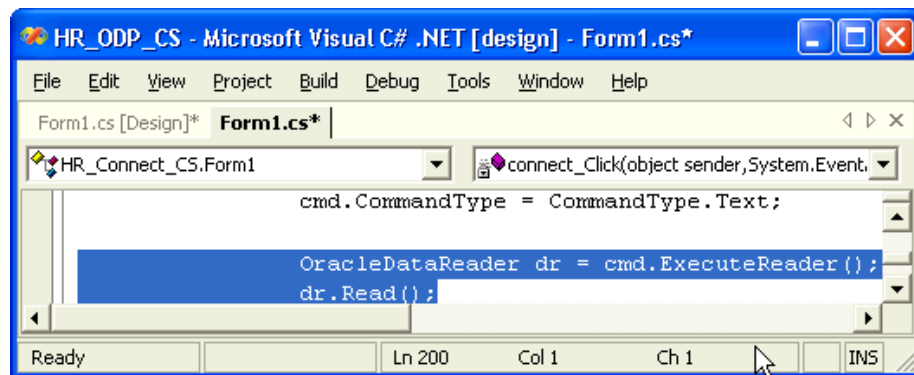
1. Run the statement using the `ExecuteReader()` method of the `OracleCommand` to return an `OracleDataReader` object, as shown in [Example 4-5](#) and [Example 4-6](#).

### Example 4-5 Starting the OracleDataReader: C#

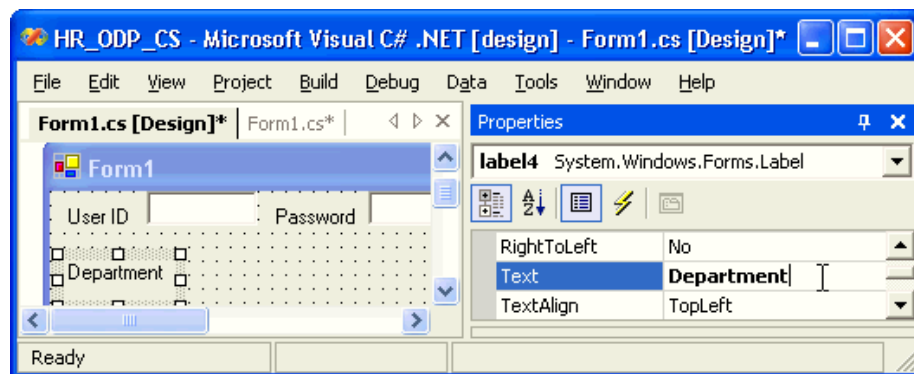
```
OracleDataReader dr = cmd.ExecuteReader();
dr.Read();
```

### Example 4-6 Starting the OracleDataReader: VB

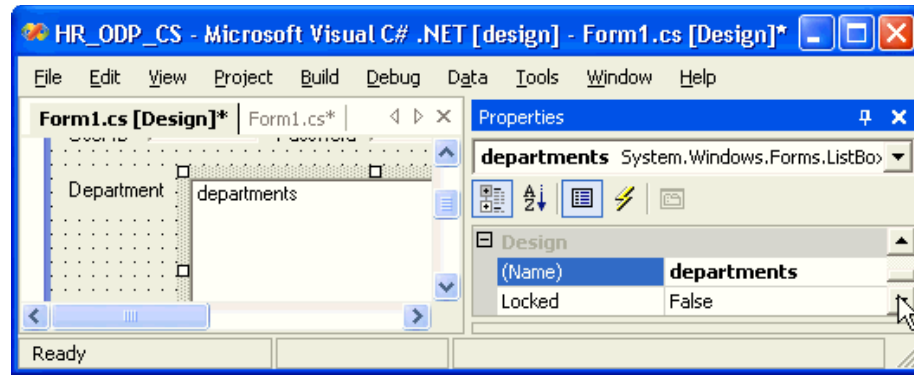
```
Dim dr As OracleDataReader = cmd.ExecuteReader()
dr.Read()
```



2. Switch to Design view.
3. From the **View** menu, select **Toolbox**.
4. From the Toolbox, under Window forms, select a **Label** and drag it onto Form1.
5. From the **View** menu, select **Properties Window**.
6. In the Properties window, change its **Text** to Department.



7. From the Toolbox, under Window forms, select a **ListBox** and drag it onto Form 1.
8. In the Properties window, under Design, change the **(Name)** to departments.



9. Close the Toolbox and the Properties window.
10. [Example 4-7](#) and [Example 4-8](#) show accessor type methods for retrieving data from the query result. There are typed accessors for returning .NET native data types, and others for returning native Oracle data types. Zero-based ordinals are passed to the accessors to specify which table column should be returned.

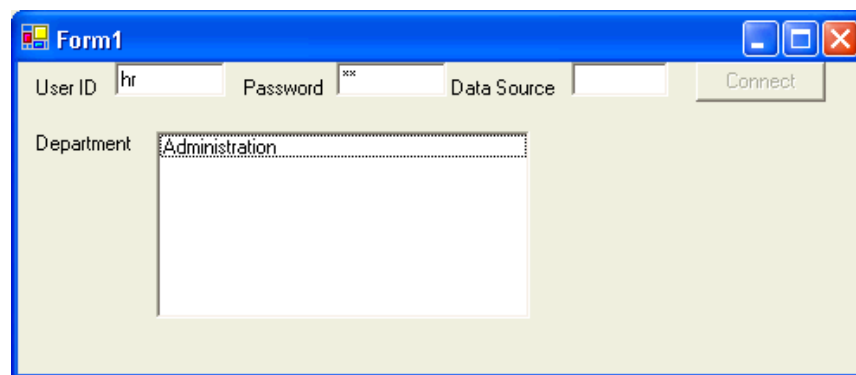
**Example 4-7 Retrieving a Value: C#**

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

**Example 4-8 Retrieving a Value: VB**

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

11. Run the application. After you connect, the departments list box shows Administration, which is the correct name for department number 10.



## Retrieving Data: Bind Variables

Bind variables are essentially placeholders in a SQL statement. When a database receives a SQL statement, it checks if the statement has already been executed and stored in memory. If the statement exists in memory, Oracle Database can reuse it and skip the task of parsing and optimizing the statement. When using bind variables, you make the statement reusable with different input values, improve query performance in the server, eliminate the need for special handling of literal quotation marks in the input, and protect against SQL injection attacks.

1. [Example 4-9](#) shows a typical `SELECT` statement that does not use bind variables, with the value 10 specified in the `WHERE` clause of the statement.

**Example 4–9 SELECT Statement without Bind Variables**

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

2. [Example 4–10](#) replaces the numerical value with a bind variable `:department_id`. The bind variable identifier always begins with a single colon, `:`, in SQL statements.

**Example 4–10 SELECT Statement with Bind Variables**

```
select department_name
from departments
where department_id = :department_id
```

3. Use the `OracleParameter` class to represent each bind variable in your .NET code. The `OracleParameterCollection` class contains the `OracleParameter` objects associated with the `OracleCommand` object for each statement. The `OracleCommand` class passes your SQL statement to the database and returns the results to your application.

You can bind variables by position or by name. The `OracleCommand` property `BindByName` (which defaults to `false`) sets the mode.

When binding by position, you must use the `Add()` method to add the parameters to the `OracleParameterCollection` in the same order as they appear in the SQL statement or stored procedure.

If you want to bind by name, you may add the parameters to the collection in any order; however, you must set the `ParameterName` property for the parameter object to the same name as the bind variable identifier in the stored procedure declaration.

4. In addition to the binding mode (by position or by name), the following properties are typically set for each parameter object: `Direction`, `OracleDbType`, `Size`, and `Value`.
  - **Direction** Bind variables may be used as output, input, or input/output parameters. The `Direction` property indicates the direction of each parameter. The default value of the `Direction` property is `Input`.
  - **OracleDbType** property indicates whether the parameter is a number, a date, a `VARCHAR2`, and so on.
  - **Size** indicates the maximum size of the data that the parameter will hold for parameters with a variable length data type, like `VARCHAR2`.
  - **Value** contains the parameter value either before statement execution (for input parameters), after execution (for output parameters), or both before and after (for input/output parameters).
5. [Example 4–11](#) and [Example 4–12](#) tie together these concepts and use a bind variable in a `SELECT` statement. Note that `Direction` property uses the default value `Input`, and the `Size` property is not set. Since the object is an input parameter, you don't need to set the `Size` property because the data provider can determine the size from the value. The changed code is in bold typeface.

**Example 4–11 Using a Bind Variable: 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 = 20;
cmd.Parameters.Add(p_department_id);

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

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

```

#### Example 4-12 Using a Bind Variable: VB

```

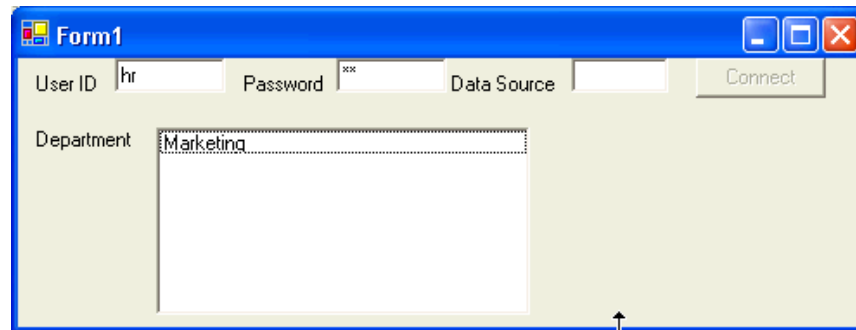
string sql = "select department_name from departments where department_id=" + _ &
    ":department_id"
Dim cmd As OracleCommand = New OracleCommand(sql, conn)
cmd.CommandType = CommandType.Text
Dim p_department_id as OracleParameter = new OracleParameter()
p_department_id.OracleDbType = OracleDbType.Decimal
p_department_id.Value = 20
cmd.Parameters.Add(p_department_id)

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

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

```

6. Run the application. After you connect, the departments list box shows Marketing, which is the correct name for department number 20.



7. Note that bind variables can also be used with UPDATE, INSERT, and DELETE statements, and also with stored procedures. [Example 4-13](#) shows how to use bind variables in an UPDATE statement; "Inserting, Deleting and Updating Data" on page 4-13 provides more details.

#### Example 4-13 UPDATE Statement with Bind Variables

```

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

```

## Retrieving Data: Multiple Values

1. A DataReader object can retrieve values for multiple columns and multiple rows. Consider a multiple column, multiple row query in [Example 4-14](#):

**Example 4-14 Querying for a Multiple Column Multiple Row Result**

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

2. A looping construct is needed to process multiple rows from the `DataReader` object. Also, a control that can display multiple rows is very useful. Because `OracleDataReader` is a forward-only, read-only cursor, it cannot be bound to an updatable or backward scrollable control such as Windows Forms `DataGrid` control. A `DataReader` is, however, compatible with a `ListBox` control, as shown in [Example 4-15](#) and [Example 4-16](#). The original code is from earlier examples in this chapter, and the changed code is in bold typeface.

**Example 4-15 Looping Through a Multi-Row Query Result: C#**

```
string sql = "select department_name from departments where department_id < 100";
OracleCommand cmd = new OracleCommand(sql, conn);
cmd.CommandType = CommandType.Text;

OracleDataReader dr = cmd.ExecuteReader();

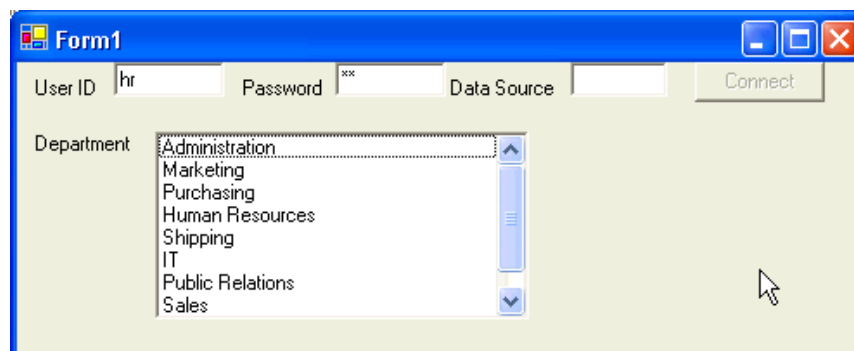
while (dr.Read())
{
    departments.Items.Add(dr.GetString(0));
}
```

**Example 4-16 Looping Through a Multi-Row Query Result: VB**

```
string sql = "select department_name from departments where department_id < 100"
Dim cmd As OracleCommand = New OracleCommand(sql, conn)
cmd.CommandType = CommandType.Text
Dim dr As OracleDataReader = cmd.ExecuteReader()

While (dr.Read())
    departments.Items.Add(dr.GetString(0))
End While
```

3. Run the application. After you connect, the departments list box shows Administration, Marketing, Purchasing, and so on, which is the correct list of department names where department number is less than 100.



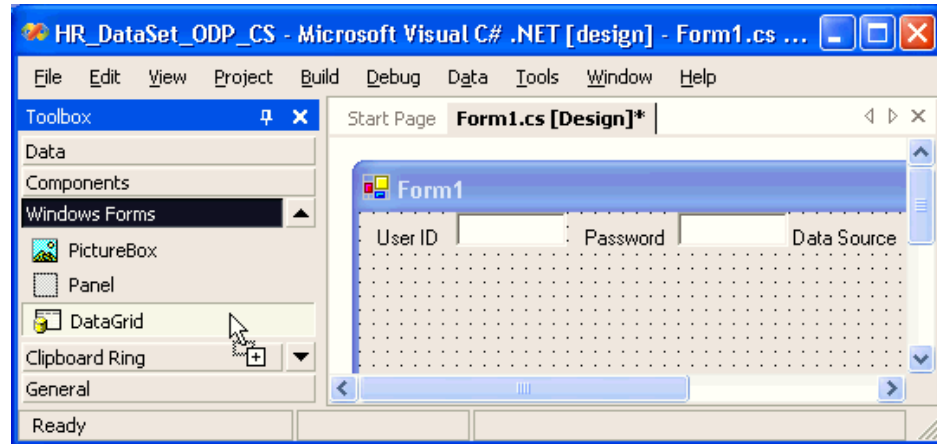
## Using the DataSet Class with Oracle Data Provider for .NET

The `DataSet` class encapsulates a memory-resident representation of data that provides a consistent relational programming model for multiple data sources. It



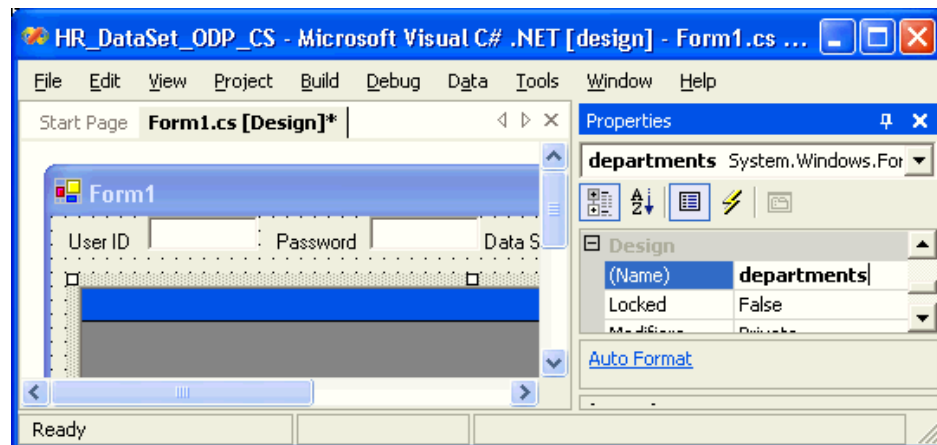
consists of one or more tables that store relational or XML data. Unlike `OracleDataReader`, a `DataSet` is updatable and backward scrollable.

1. Follow the steps in section "Copying a Project" on page 4-1 to create a new copy of the `HR_Connect_CS` project (`HR_Connect_VB` for Visual Basic). Name the new project `HR_DataSet_ODP_CS` (`HR_DataSet_ODP_VB` for Visual Basic).
2. Switch to design view ( use **Shift+F7** keyboard shortcut).
3. From the **View** menu, select **Toolbox**.
4. From the Toolbox, under Windows Forms, select a **Data Grid** and drag it onto `Form1`.



A data grid appears. Expand the data grid, and close the Toolbox.

5. Right-click the data grid graphical element, and select **Properties**. In the properties list, under Design, change **(Name)** to `departments`.



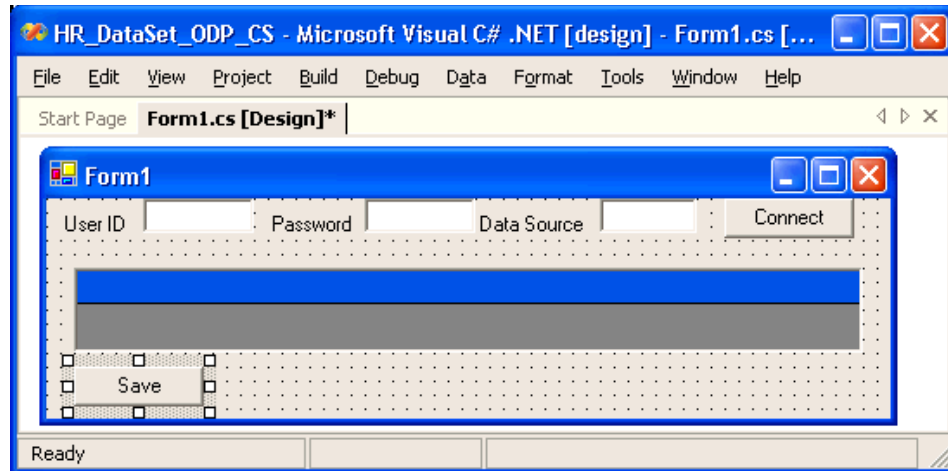
Close the Properties window.

6. From the **View** menu, select **Toolbox**.
7. From the Toolbox, under Windows Forms, drag and drop a **Button** onto `Form1`.
8. Right-click the new button, and select **Properties**.
9. In the Properties window, under Appearance, change **Text** to `Save`.

Under Design, change (**Name**) to save.

Click the lightning icon (events), and then click the highlighted **Click** event. From the drop-down window, select `save_Click`.

Close the Properties window. Switch to code view using the **F7** keyboard shortcut.



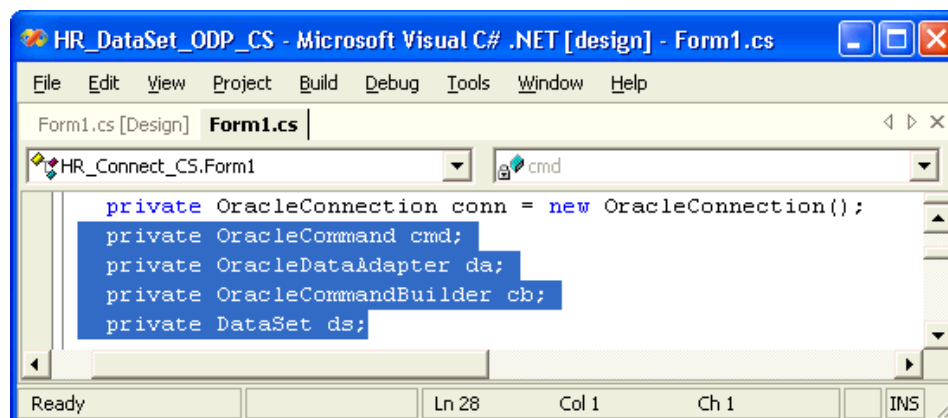
10. Add variable declarations in [Example 4-17](#) and [Example 4-18](#) to the class variables.

**Example 4-17 Using DataSet Class: Declaring Variables: C#**

```
private OracleCommand cmd;
private OracleDataAdapter da;
private OracleCommandBuilder cb;
private DataSet ds;
```

**Example 4-18 Using DataSet Class: Declaring Variables: VB**

```
Private cmd As OracleCommand
Private da As OracleDataAdapter
Private cb As OracleCommandBuilder
Private ds As DataSet
```



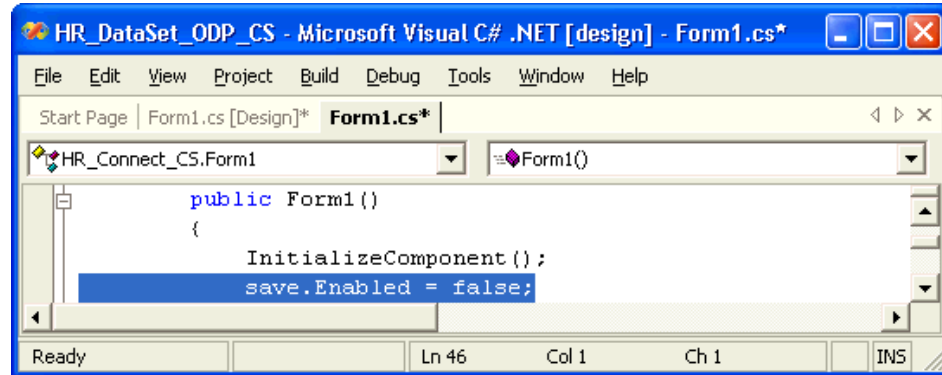
11. Within the `Form()` method, add the code shown in [Example 4-19](#) and [Example 4-20](#).

**Example 4-19 Disabling the Save Button: C#**

```
save.Enabled = false;
```

**Example 4-20 Disabling the Save Button: VB**

```
save.Enabled = false
```



12. Within the `connect_Click()` method try block, as shown in [Example 4-21](#) and [Example 4-22](#),

- query the database
- fill the `DataSet` with the result of the command query
- bind the `DataSet` to the data grid
- enable the **Save** button

The changed code is in bold typeface.

**Example 4-21 Binding Data to the Grid: 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];

save.Enabled = true;
```

**Example 4-22 Binding Data to the Grid: VB**

```
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()

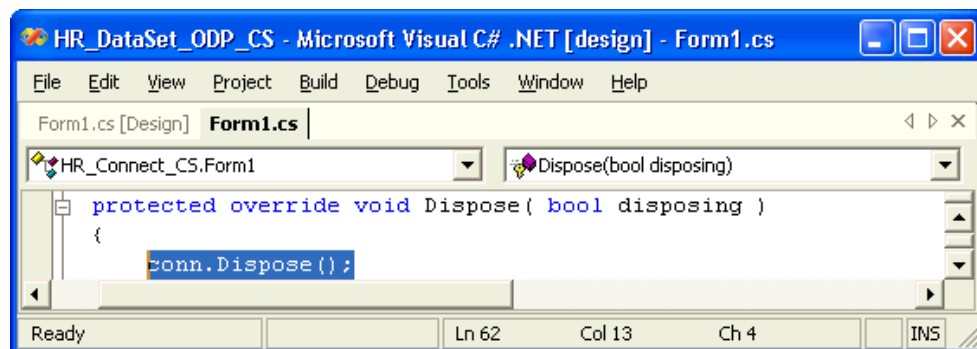
da.Fill(ds)

departments.DataSource = ds.Tables[0]

save.Enabled = true

```

13. The finally block in the `connect_Click()` method contains code for disposing the connection, a `conn.Dispose()` call. Move this call to the top of the general `Dispose()` method. This is necessary to keep the connection open after the query result returns, so that data changes made by the end user are propagated to the database.



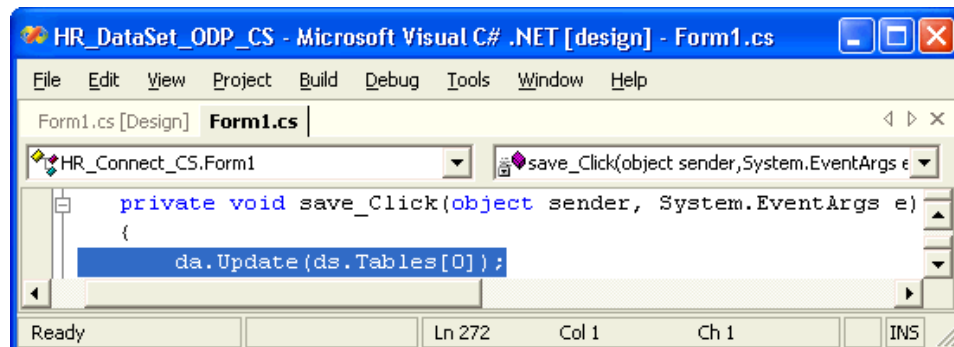
14. Your code also contains a new `save_Click()` method created in Step 9. Add there the code for updating the data, shown in [Example 4-23](#) and [Example 4-24](#).

**Example 4-23 Updating DataSet: C#**

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

**Example 4-24 Updating DataSet: VB**

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



15. Save Form1 using **Ctrl+S** keyboard shortcut.
16. Run the application using the **F5** keyboard shortcut.

- After you successfully connect to the database, the data grid is populated with the results of the query.

Form1

User ID: hr Password: \*\* Data Source: [ ] Connect

Department

	DEPARTMEN	DEPARTMENTEN	MANAGER_I	LOCATION_I
▶	10	Administration	200	1700
	20	Marketing	201	1800
	30	Purchasing	114	1700
	40	Human Reso	203	2400
	50	Shipping	121	1500
*				

Save

## Inserting, Deleting and Updating Data

- At the bottom of the data grid, enter a new record at the \* prompt:
  - For DEPARTMENT\_ID, enter 5
  - For DEPARTMENT\_NAME, enter Community Outreach
  - Leave MANAGER\_ID as null
  - For LOCATION\_ID, enter 1700

Click the **Save** button.

Form1

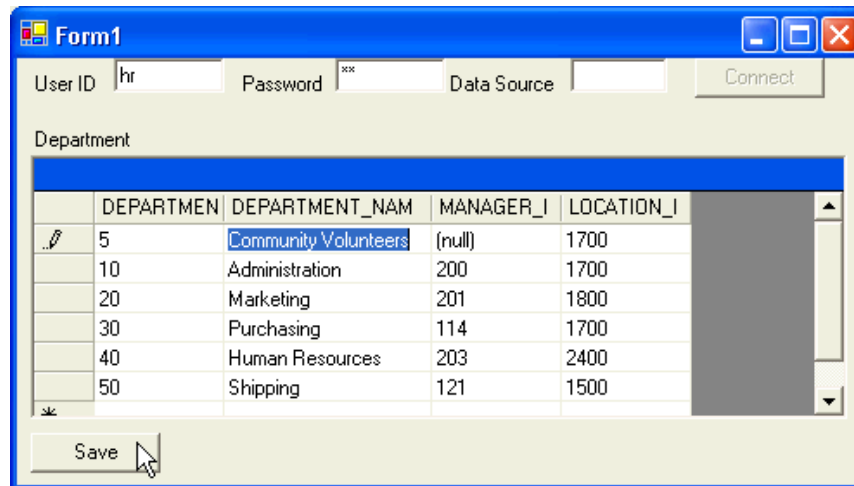
User ID: hr Password: \*\* Data Source: [ ] Connect

Department

	DEPARTMEN	DEPARTMENT_NAM	MANAGER_I	LOCATION_I
	10	Administration	200	1700
	20	Marketing	201	1800
	30	Purchasing	114	1700
	40	Human Resources	203	2400
	50	Shipping	121	1500
✎	5	Community Outreach	(null)	1700
*				

Save

- To check if the new record is saved, close the application, and start it again using the F5 keyboard shortcut.
- Connect to the database, and note that the new department is now part of the DEPARTMENTS table.
- Change the name of the department to Community Volunteers, and click the **Save** button.



Form1

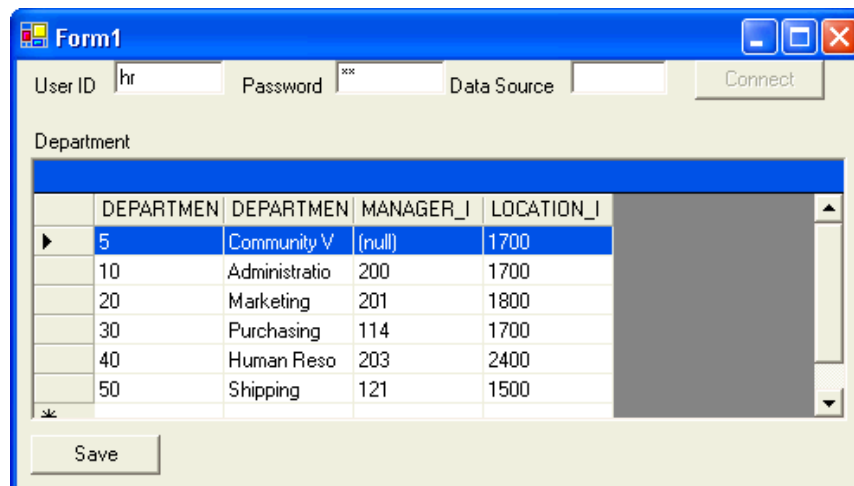
User ID: hr Password: xxx Data Source: [ ] Connect

Department

	DEPARTMEN	DEPARTMENT_NAM	MANAGER_I	LOCATION_I
	5	Community Volunteers	(null)	1700
	10	Administration	200	1700
	20	Marketing	201	1800
	30	Purchasing	114	1700
	40	Human Resources	203	2400
	50	Shipping	121	1500

Save

- Repeat Step 2, connect to the database, and note that the name of the department is changed.
- Select the entire record you just changed (click the cursor icon before it), and delete it using the **Delete** key. Click the **Save** button.



Form1

User ID: hr Password: xxx Data Source: [ ] Connect

Department

	DEPARTMEN	DEPARTMEN	MANAGER_I	LOCATION_I
	5	Community V	(null)	1700
	10	Administratio	200	1700
	20	Marketing	201	1800
	30	Purchasing	114	1700
	40	Human Reso	203	2400
	50	Shipping	121	1500

Save

- Repeat Step 2, connect to the database, and note that the name of the new record is no longer part of the DEPARTMENTS table.
- Close the application.

---

## Using Oracle Developer Tools for Visual Studio .NET

This chapter explains how to use Oracle Developer Tools.

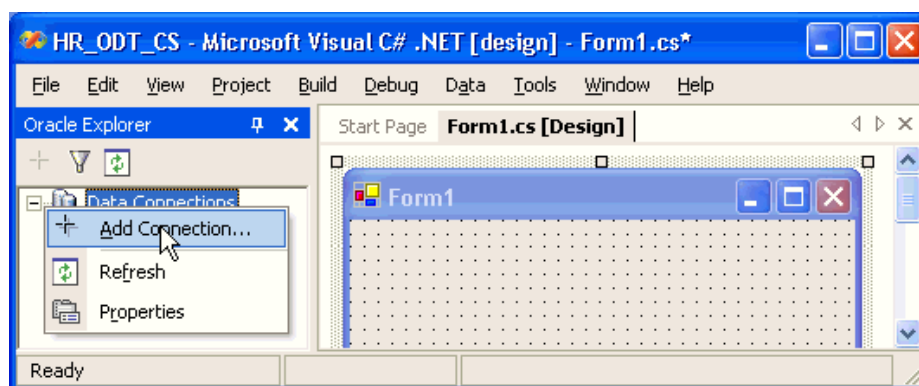
This chapter contains the following sections:

- [Connecting to the Oracle Database](#)
- [Creating a Table and Its Columns](#)
- [Creating a Table Index](#)
- [Adding Table Constraints](#)
- [Adding Data to a Table](#)
- [Generating Code Automatically](#)
- [Enabling Database Updates](#)

### Connecting to the Oracle Database

To connect to an Oracle Database from Visual Studio .NET, follow these steps:

1. Follow instructions in Section ["Starting a New Project"](#) on page 3-1. Name the C# project HR\_ODT\_CS. If starting a VB project, name it HR\_ODT\_VB.
2. From the **View** menu, select **Oracle Explorer**.
3. In Oracle Explorer, right-click **Data Connections**. From the menu, select **Add Connection**.



The application opens an Add Connection window.

4. In the Add Connection window, enter the following information:

**Data source name:** Use the `Local Database` if you are connecting to a database on the same machine. Otherwise, use the alias of the remote database instance.

Select the **Use a specific user name and password** option.

For **User name**, enter `hr`.

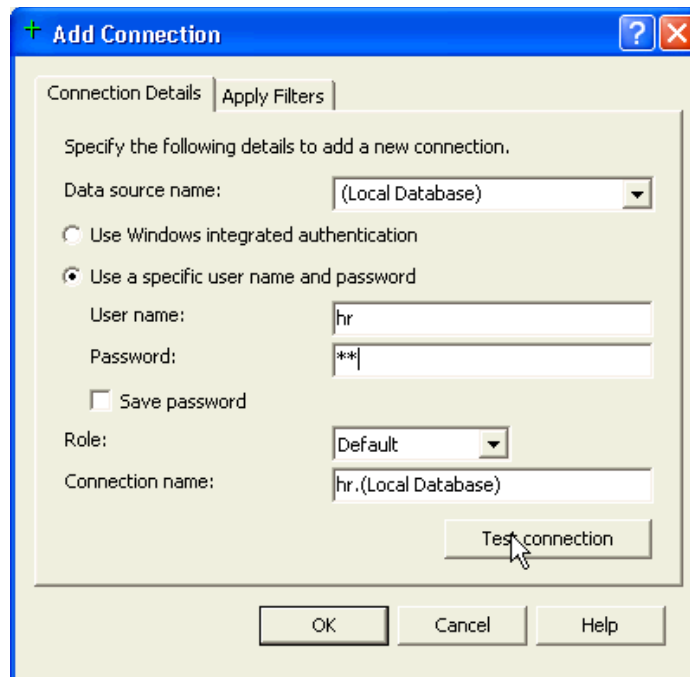
For **Password**, enter `hr`, or the password that was set when unlocking and setting up the `hr` account.

To save the password for future sessions, check the **Save password** box.

Ensure that **Role** is set to `Default`. This refers to the default roles that have been granted to the user `hr`.

The **Connection name** should be generated automatically from the **Data source name** and the **User name** values.

Click the **Apply Filters** tab, and check that the `HR` schema is in the **Displayed schemas** column. Only the schema objects (tables, views, and so on) from the schemas selected in the **Apply Filters** tab are displayed when you expand the schema category nodes in the data connection.



Click the **Connection Details** tab, and then click **Test connection**.

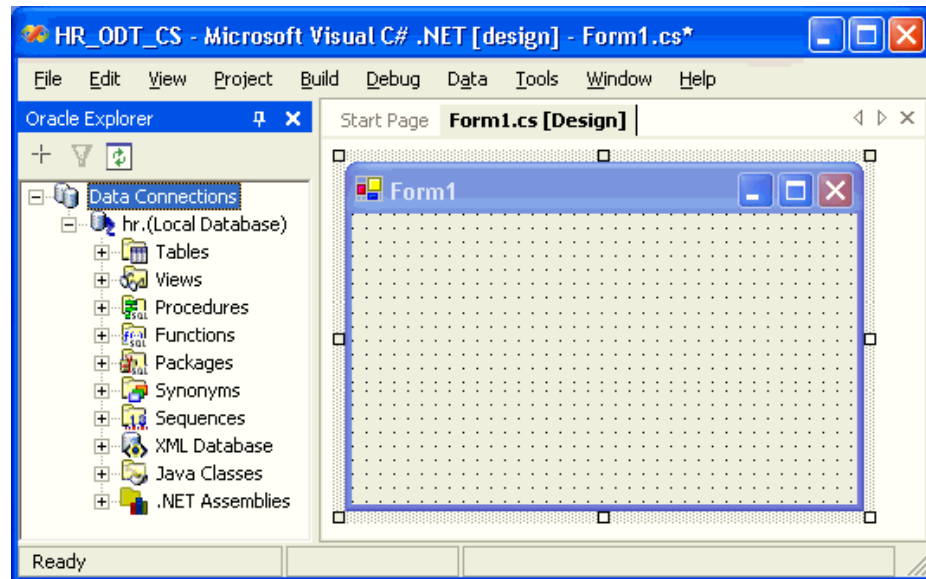


The test should succeed. Click **OK**.



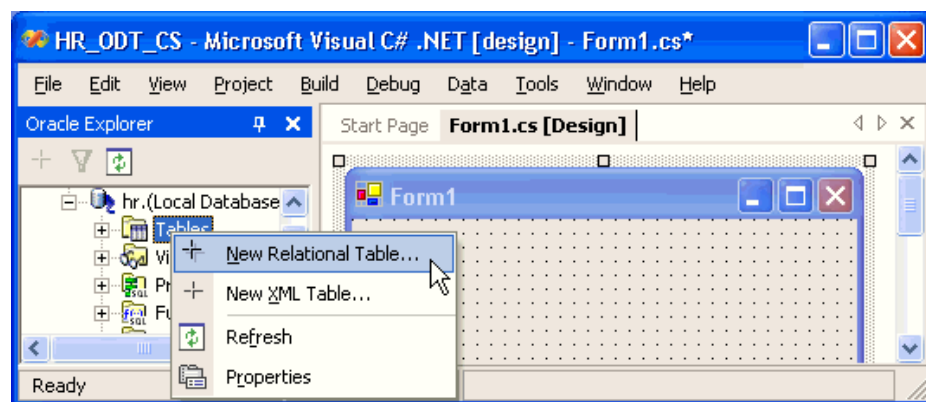
If the test fails, it may be due to one or more of the following issues that you must address before proceeding with further steps:

- The database is not started.
  - The database connectivity is not properly configured.
  - You do not have the correct user name, password, and role.
5. Oracle Explorer should now contain the hr . (Local Database) connection. Expand the connection to show the contents of the hr schema. You should see Tables, Views, Procedures, Functions, and so on.



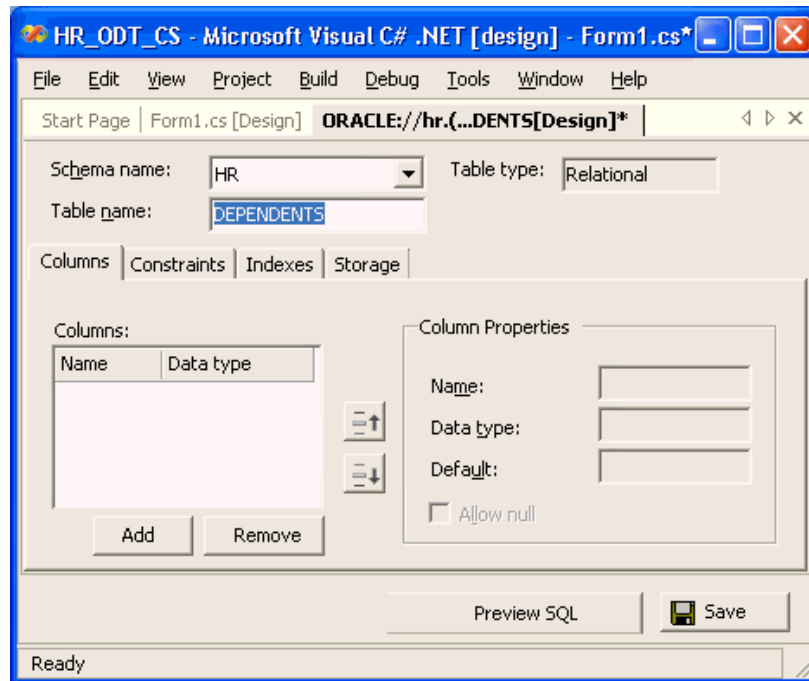
## Creating a Table and Its Columns

1. In Oracle Explorer, right-click **Tables** and select **New Relational Table**.

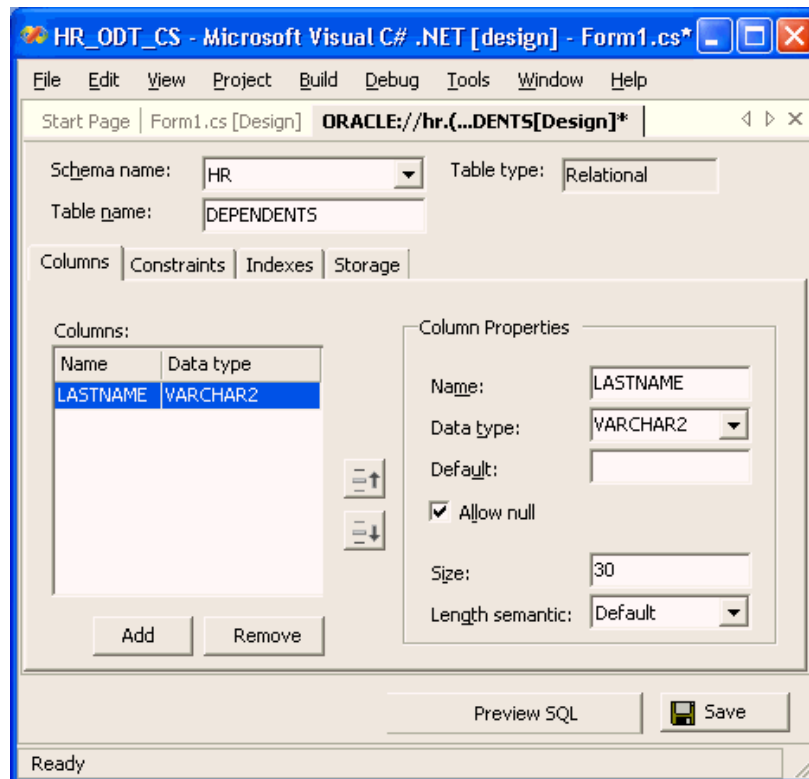


A table design window appears.

2. In design view, enter `DEPENDENTS` for **Table name**.



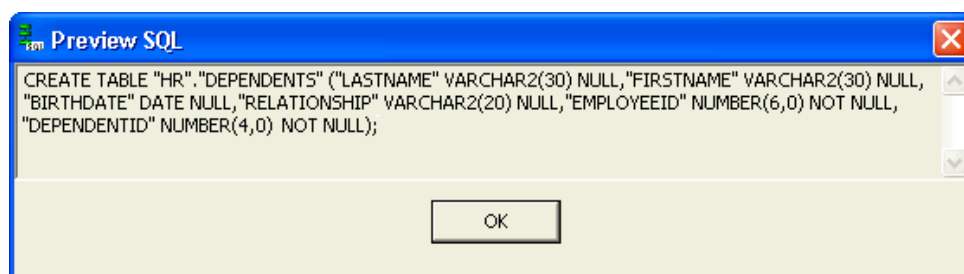
3. Add the LASTNAME column. Click the **Add** button under the Columns area. Under Column Properties, enter **Name** LASTNAME, **Data Type** VARCHAR2, and **Size** 30. Leave all other properties at their default values.



4. Add the `FIRSTNAME` column. Click the **Add** button under the Columns area. Under Column Properties, enter **Name** `FIRSTNAME`, **Data Type** `VARCHAR2`, and **Size** 30. Leave all other properties at their default values.
5. Add the `BIRTHDATE` column. Click the **Add** button under the Columns area. Under Column Properties, enter **Name** `BIRTHDATE`, **Data Type** `DATE`, and leave all other properties at their default values.
6. Add the `RELATIONSHIP` column. Click the **Add** button under the Columns area. Under Column Properties, enter **Name** `RELATIONSHIP`, **Data Type** `VARCHAR2`, and **Size** 20. Leave all other properties at their default values.
7. Add the `EMPLOYEEID` column. Click the **Add** button under the Columns area. Under Column Properties, enter **Name** `EMPLOYEEID`, **Data Type** `NUMBER`, deselect **Allow null**, enter **Precision** 6 and **Scale** 0.
8. Add the `DEPENDENTID` column. Click the **Add** button under the Columns area. Under Column Properties, enter **Name** `DEPENDENTID`, **Data Type** `NUMBER`, deselect **Allow null** check box, enter **Precision** 4 and **Scale** 0.
9. Click **Preview SQL**. The SQL statement for constructing the table, as shown in [Example 5-1](#), appears in the Preview SQL window.

**Example 5-1 Generated SQL Form of the New Table**

```
CREATE TABLE "HR"."DEPENDENTS" ("LASTNAME" VARCHAR2(30) NULL,
  "FIRSTNAME" VARCHAR2(30) NULL, "BIRTHDATE" DATE NULL,
  "RELATIONSHIP" VARCHAR2(20) NULL, "EMPLOYEEID" NUMBER(6,0) NOT NULL,
  "DEPENDENTID" NUMBER(4,0) NOT NULL);
```



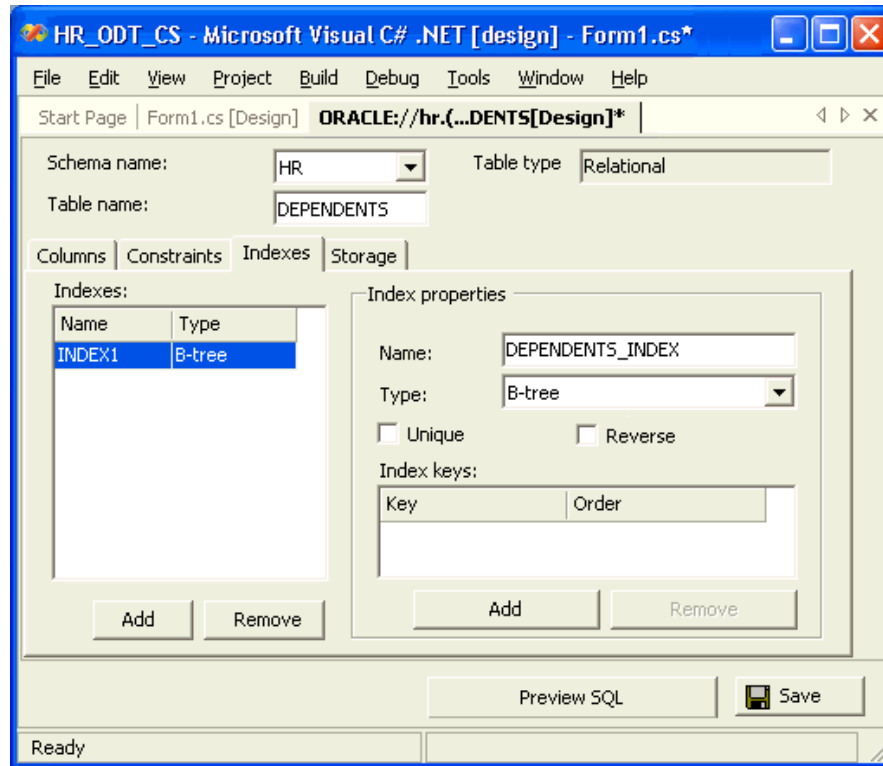
Click **OK** to close the Preview SQL window.

10. In the table design view, click **Save**.

## Creating a Table Index

Now you must create an index for the `DEPENDENTS` table.

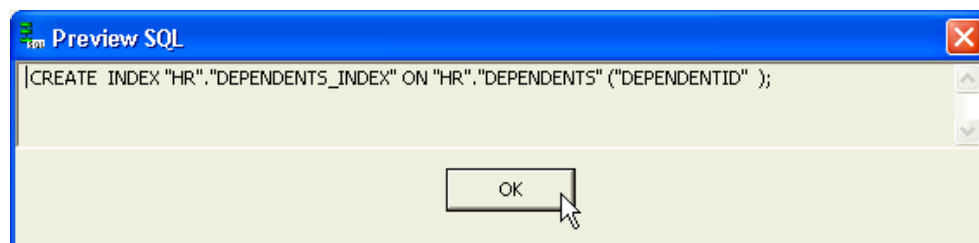
1. In the design view, click the **Indexes** tab.
2. Click the **Add** button under the Indexes area. Under Index properties, enter **Name** `DEPENDENTS_INDEX`, and leave all other properties in their default state.



3. In the Index properties area, click **Add**.
4. Under Index keys, from the **Key** column, select **DEPENDENTID** from the drop-down list.
5. Click **Preview SQL**. A Preview SQL window appears, displaying the SQL statement that constructs the index, as shown in [Example 5-2](#).

#### Example 5-2 Creating a Table Index in SQL

```
CREATE INDEX "HR"."DEPENDENTS_INDEX" ON "HR"."DEPENDENTS" ("DEPENDENTID" );
```



Click **OK** to close the Preview SQL window.

6. In the table design view, click **Save**.

## Adding Table Constraints

Now you must add constraints to the new table.

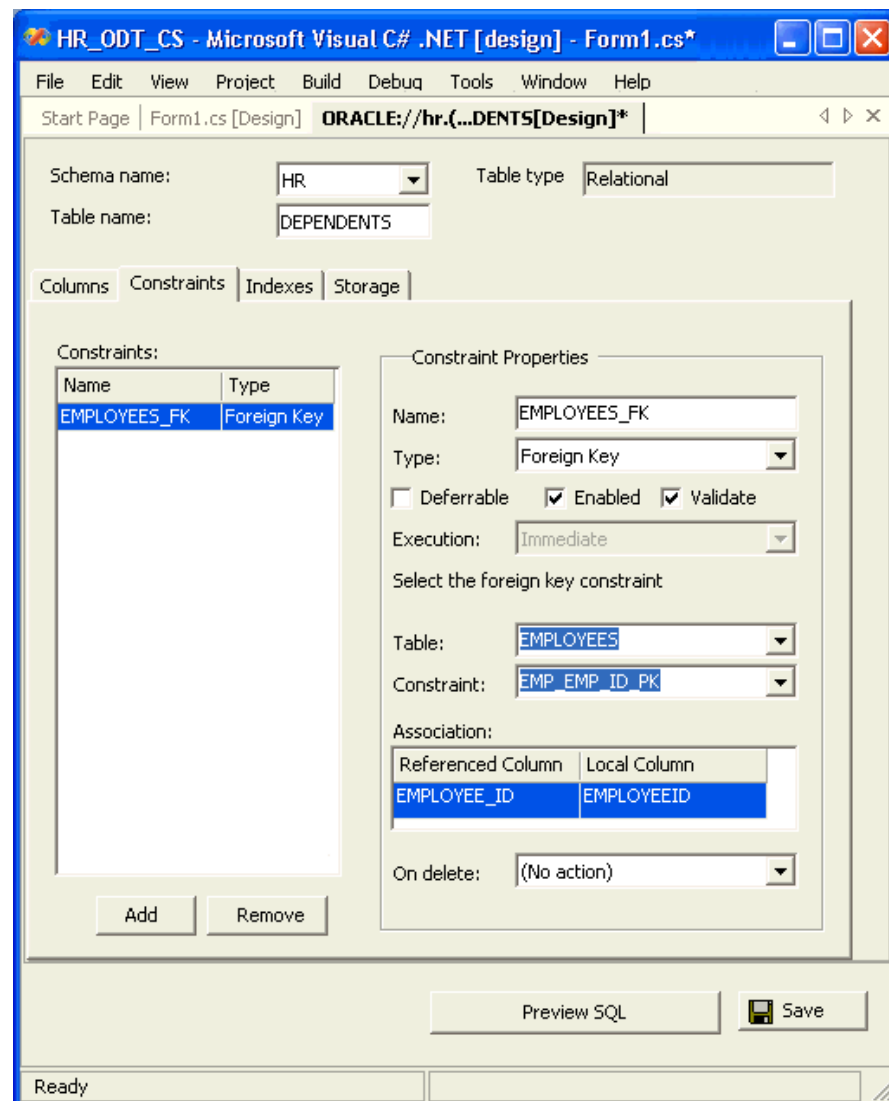
1. To create a foreign key to the EMPLOYEES table, click the **Constraints** tab. Note that depending on your configuration, there may already be default check constraints in the list.

Under the Constraints area, click **Add**.

Under Constraint Properties, enter **Name** EMPLOYEES\_FK, select **Type** Foreign Key from the drop-down list, select **Table** EMPLOYEES and **Constraint** EMP\_EMP\_ID\_PK.

Under Association, select EMPLOYEE\_ID as **Referenced Column** and EMPLOYEEID as **Local Column**.

Leave all other properties at their default values.



2. To create a primary key for the new table, DEPENDENTS, under the Constraints area click **Add**.

Under Constraint Properties, enter **Name** DEPENDENTS\_PK and select **Type** Primary Key from the drop-down list.

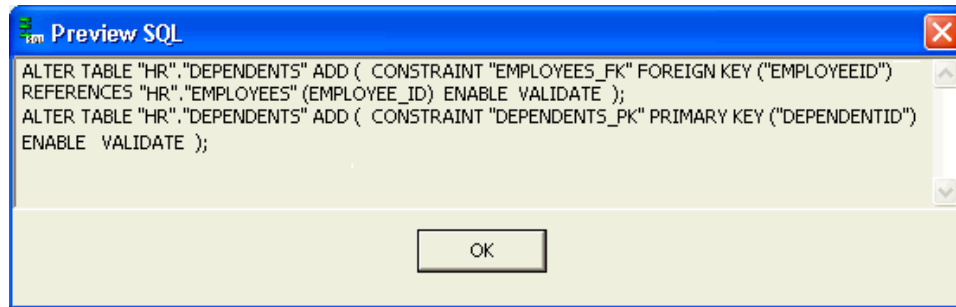
Under Constraint Properties, click **Add**.

Under Primary Key Columns, select `DEPENDENTID`. Leave all other properties at their default values.

3. Click **Preview SQL**. A Preview SQL window appears. [Example 5-3](#) shows the code generated for constraints on table `DEPENDENTS`. Note that addition of both constraints is an `ALTER TABLE` command, because the constraints change the definitions of columns `DEPENDENTID` and `EMPLOYEEID`.

**Example 5-3 Adding Foreign Key and Primary Key Constraints to a Table**

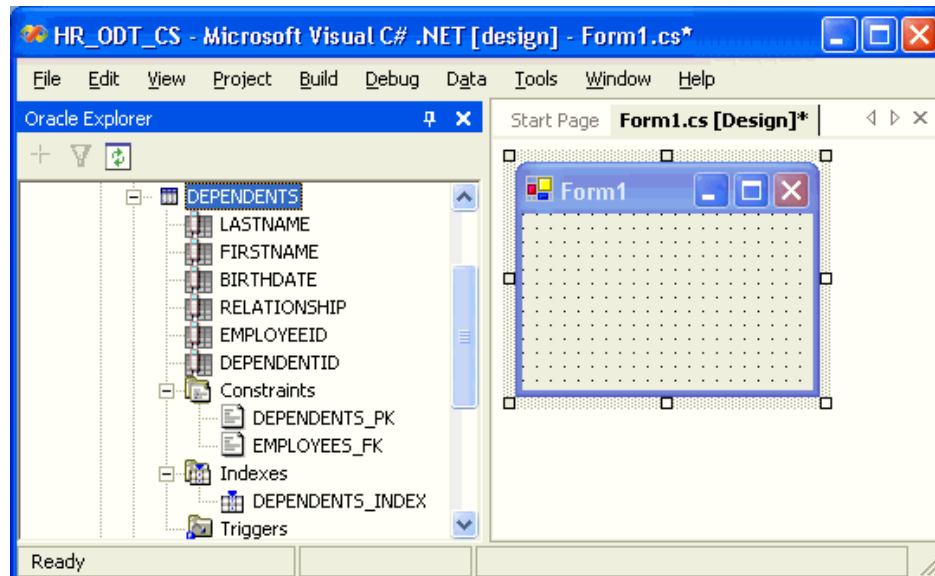
```
ALTER TABLE "HR"."DEPENDENTS" ADD ( CONSTRAINT "EMPLOYEES_FK" FOREIGN KEY
("EMPLOYEEID") REFERENCES "HR"."EMPLOYEES" (EMPLOYEE_ID) ENABLE VALIDATE );
ALTER TABLE "HR"."DEPENDENTS" ADD ( CONSTRAINT "DEPENDENTS_PK" PRIMARY KEY
("DEPENDENTID") ENABLE VALIDATE );
```



Click **OK** to close the Preview SQL window.

4. In the table design view, click **Save**.

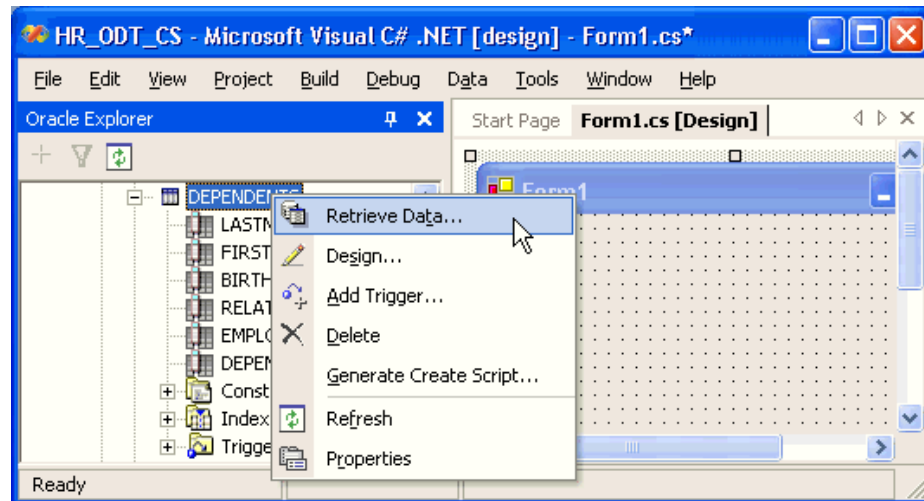
Notice that if you expand the `DEPENDENTS` table in the Oracle Explorer, all the columns, constraints and indexes of the table are visible.



## Adding Data to a Table

You must now add data to the new `DEPENDENTS` table.

1. In Oracle Explorer, right-click the `DEPENDENTS` table and select **Retrieve Data**.

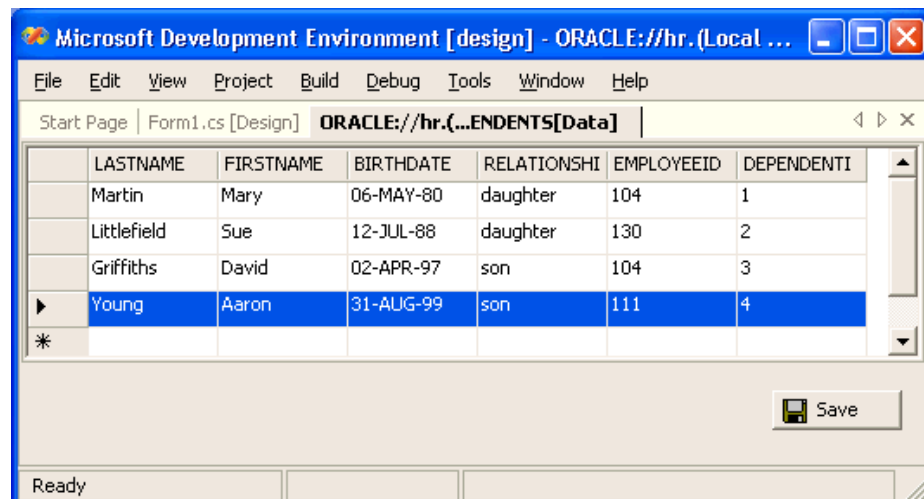


A table grid for DEPENDENTS appears in design view.

2. Enter the four records listed in [Table 5-1](#) into the table grid.

**Table 5-1 New Data for the DEPENDENTS Table**

LASTNAME	FIRSTNAME	BIRTHDATE	RELATIONSHIP	EMPLOYEEID	DEPENDENTID
Martin	Mary	06-MAY-80	daughter	104	1
Littlefield	Sue	12-JUL-88	daughter	130	2
Griffiths	David	02-APR-97	son	104	3
Young	Aaron	31-AUG-99	son	111	4



Note that the data is automatically saved as you move between rows.

## Generating Code Automatically

To explore the content of table DEPARTMENTS, we will build a form that uses a simple table query.

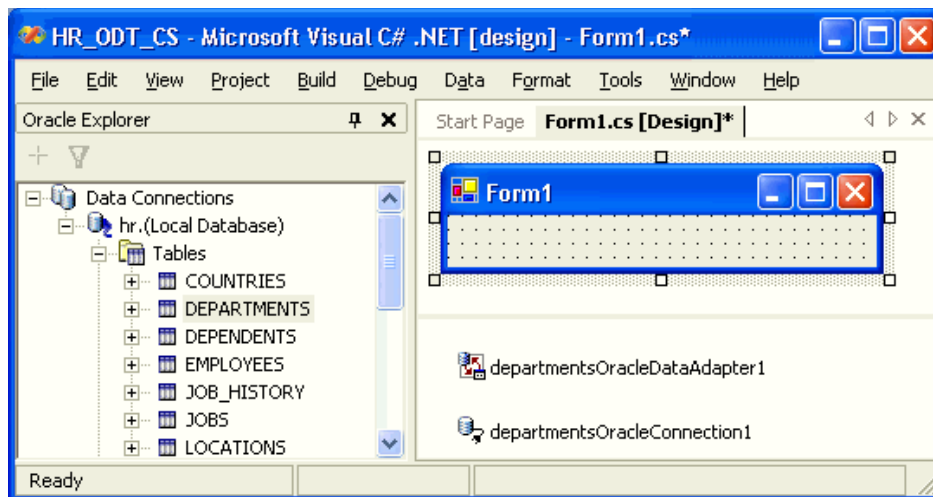
1. Switch to the Form1 design view (**Shift+F7** keyboard shortcut).
2. In Oracle Explorer, expand **Data Connections**, expand **hr. (Local Database)**, and finally expand the **Tables** component.

Using your mouse, select the **DEPARTMENTS** table. Drag and drop the table onto Form1 in the **Designer** window.

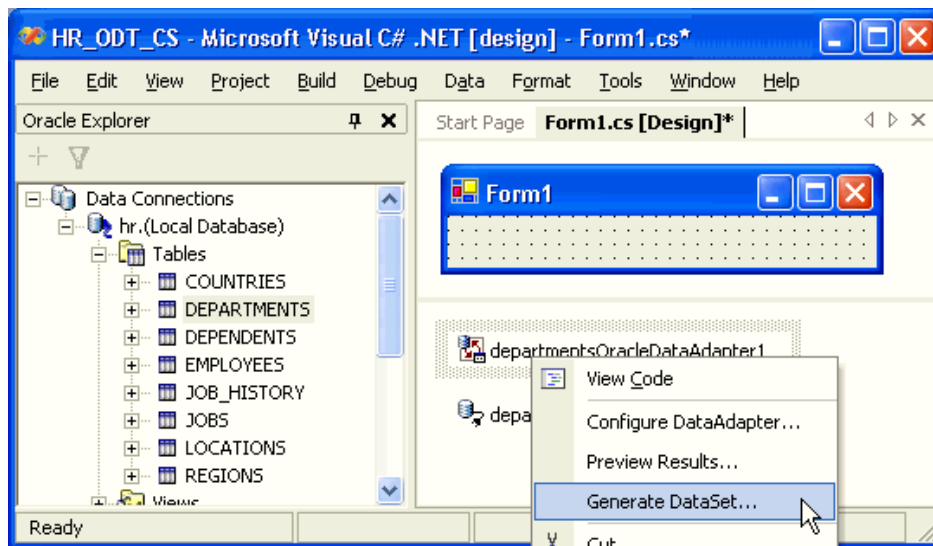
A Microsoft Development Environment pop-up window will ask if you wish to save the connection password in the generated code. While it is not advisable to save connection password within your form in clear text for security reasons, we will do it in this demonstration. Click **Yes**.

You will notice that this action creates an `OracleConnection` object, `departmentsOracleConnection1`, and an `OracleDataAdapter` object, `departmentsOracleDataAdapter1`. Both appear under the Design window.

These objects represent automatically generated code for Form1.

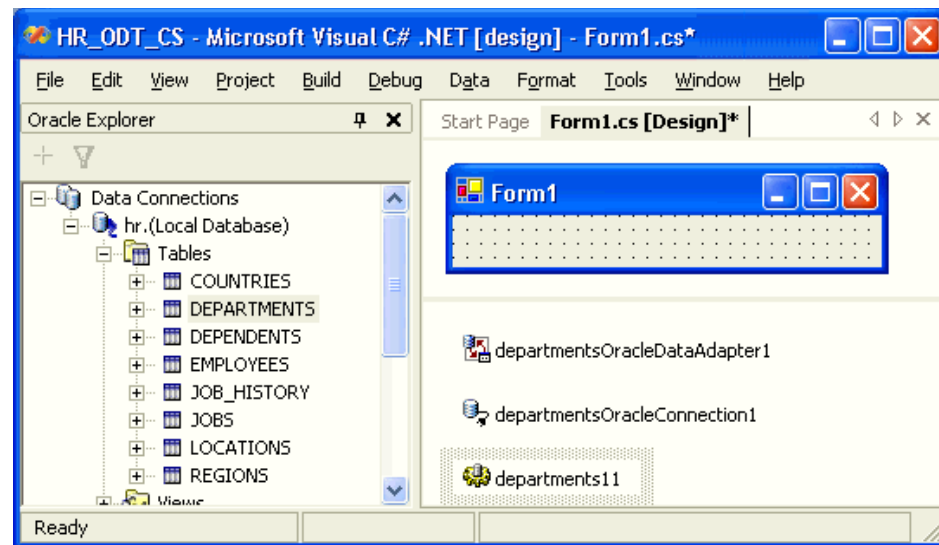


3. Right-click the `departmentsOracleDataAdapter1`, and select **Generate DataSet**.

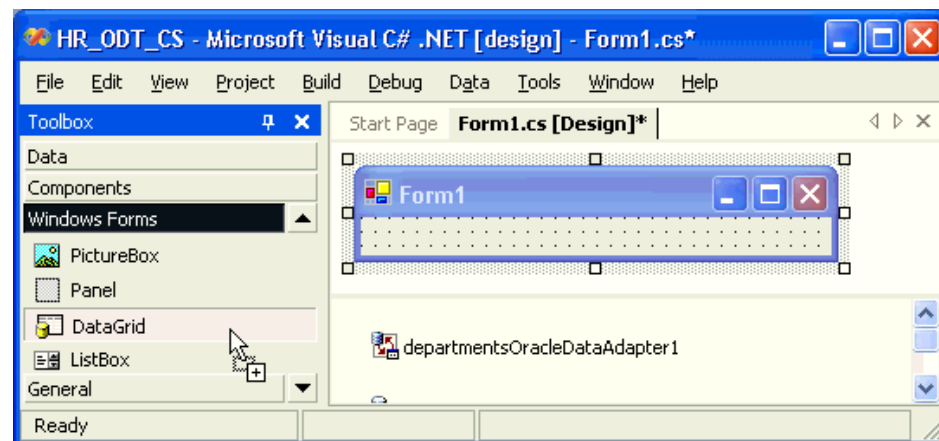




4. The object `departments11` is now added to your Design window.



5. From the Toolbox, under Window Forms, select **DataGrid** and drag it onto Form1.

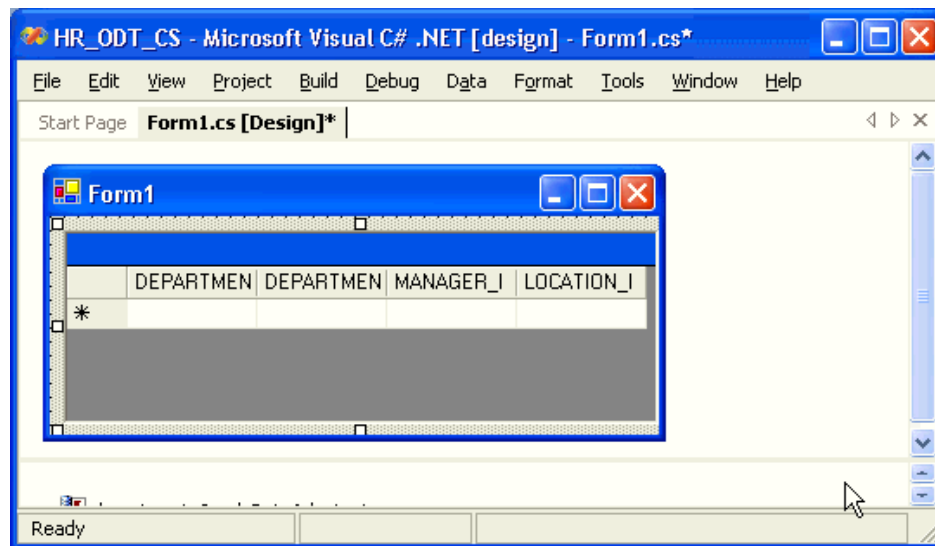


Enlarge both **Form1** and the **DataGrid**.

6. Right-click the **DataGrid** and select **Properties**.
7. In the Properties window, under **Data**, set the **DataSource** parameter to `departments11.Departments` from the drop-down list.

Close the Properties window.

The **DataGrid** now contains the column headings from the table `DEPARTMENTS`.



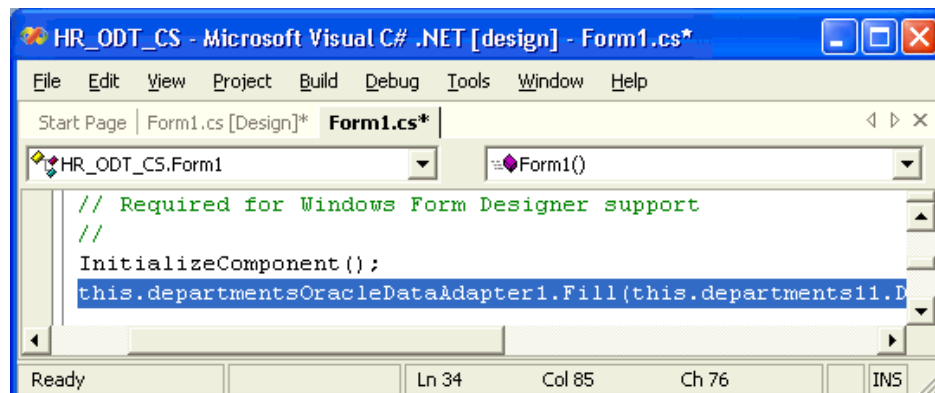
8. Switch to code view by using the F7 keyboard shortcut.
9. Immediately after the `InitializeComponent()` ; command, add the code in [Example 5-4](#) or [Example 5-5](#).

**Example 5-4 Filling Data into the Form: C#**

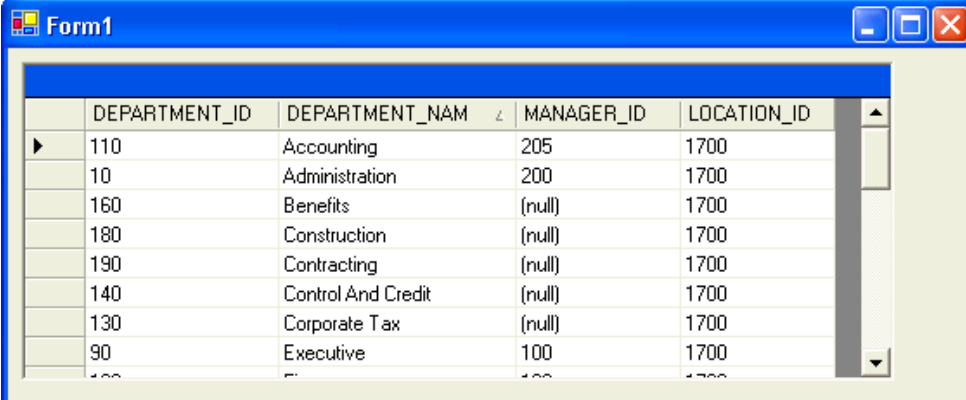
```
this.departmentsOracleDataAdapter1.Fill(this.departments11.Departments);
```

**Example 5-5 Filling Data into the Form: VB**

```
Me.departmentsOracleDataAdapter1.Fill(Me.departments11.Departments)
```



10. Run the application (use the F5 keyboard shortcut).

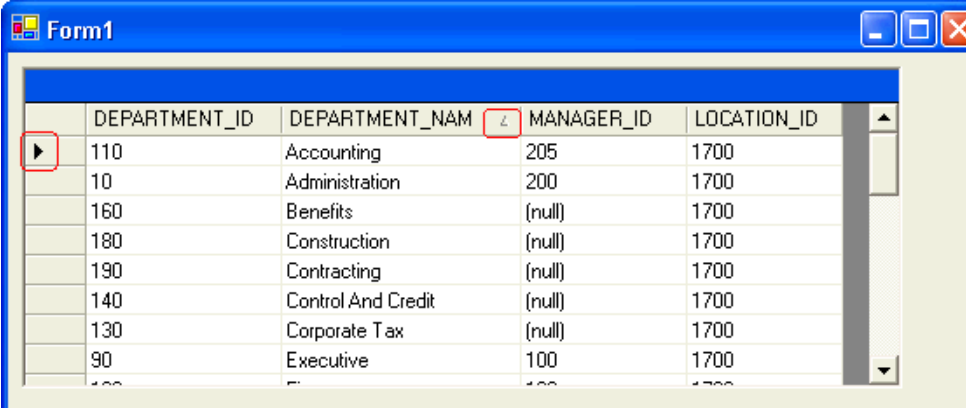


	DEPARTMENT_ID	DEPARTMENT_NAM	MANAGER_ID	LOCATION_ID
▶	110	Accounting	205	1700
	10	Administration	200	1700
	160	Benefits	(null)	1700
	180	Construction	(null)	1700
	190	Contracting	(null)	1700
	140	Control And Credit	(null)	1700
	130	Corporate Tax	(null)	1700
	90	Executive	100	1700

You may need to increase the width of the columns to see all the data.

11. You can navigate across data returned by the application by clicking the mouse down the data set; the current record will be marked by the cursor.

You can also sort the records, either in ascending or descending order, on any of the columns, by clicking on the column heading (notice the direction indicator in the DEPARTMENT\_NAME column).



	DEPARTMENT_ID	DEPARTMENT_NAM	MANAGER_ID	LOCATION_ID
▶	110	Accounting	205	1700
	10	Administration	200	1700
	160	Benefits	(null)	1700
	180	Construction	(null)	1700
	190	Contracting	(null)	1700
	140	Control And Credit	(null)	1700
	130	Corporate Tax	(null)	1700
	90	Executive	100	1700

12. Close the application.

## Enabling Database Updates

1. From Section ["Using the DataSet Class with Oracle Data Provider for .NET"](#) on page 4-8, follow Steps 6 through 9 to create a **Save** button.

2. Double-click **Save**.

The code view appears, with focus on the new and empty `save_Click()` method.

3. Change the code of the `save_Click()` method to bind the table update event to the button, as shown in [Example 5-6](#) and [Example 5-7](#).

### **Example 5-6 The `save_Click()` Method: C#**

```
private void save_Click(object sender, System.EventArgs e)
{
```

```
        departmentsOracleDataAdapter1.Update(departments11);  
    }
```

**Example 5-7 The save\_Click() Method: VB**

```
Private Sub save_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles Save.Click  
    departmentsOracleDataAdapter1.Update(departments11)  
End Sub
```

4. Run the application (F5 keyboard shortcut).  
Your Form1 application window should appear.
5. To use the application, follow the instructions in Section ["Inserting, Deleting and Updating Data"](#) on page 4-13.

---

## Using PL/SQL Stored Procedures and REF Cursors

This chapter contains the following sections:

- [Introduction to PL/SQL Packages and Package Bodies](#)
- [Introduction to PL/SQL Stored Procedures](#)
- [Introduction to Ref Cursors](#)
- [Creating a PL/SQL Stored Procedure that Uses Ref Cursors](#)
- [Running a PL/SQL Stored Procedure Using Oracle Data Provider for .NET](#)

### Introduction to PL/SQL Packages and Package Bodies

A PL/SQL package stores related items as a single logical entity. A package is composed of two distinct pieces:

- The **package specification** defines what is contained in the package; it is analogous to a header file in a language such as C++. The specification defines all public items. The specification is the published interface to a package.
- The **package body** contains the code for the procedures and functions defined in the specification, and the code for private procedures and functions that are not declared in the specification. This private code is only "visible" within the package body.

The package specification and body are stored as separate objects in the data dictionary and can be seen in the `user_source` view. The specification is stored as the `PACKAGE` type, and the body is stored as the `PACKAGE BODY` type.

While it is possible to have a specification without a body, as when declaring a set of public constants, it is not possible to have a body with no specification.

### Introduction to PL/SQL Stored Procedures

A stored procedure is a named set of PL/SQL statements designed to perform an action. Stored functions have a single return value parameter. Unlike functions, procedures may or may not return values.

### Introduction to Ref Cursors

Ref cursors are one of the most powerful, flexible, and scalable methods for returning query results from an Oracle Database to a client application.

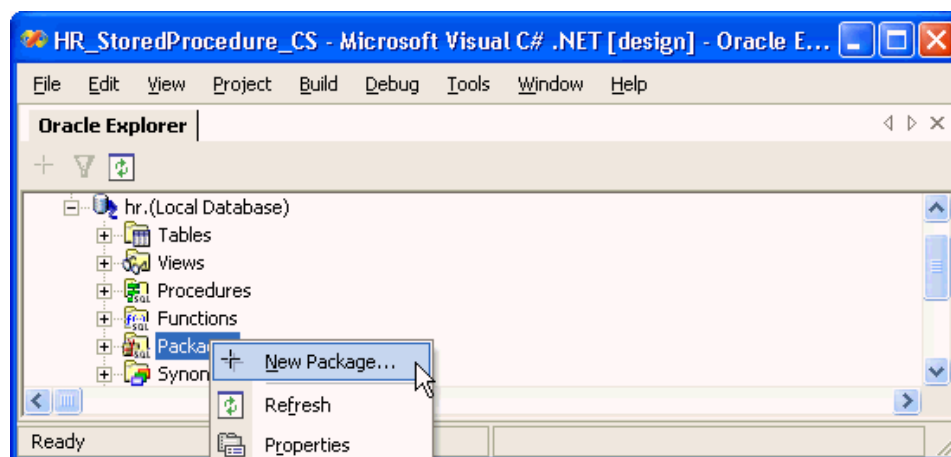
A ref cursor is a PL/SQL datatype whose value is the address of the memory location of a query work area on the database server. A query work area can be thought of as the result set, or a row set, on the server; it is the location where the results of a query are stored in server memory. In essence, a ref cursor is a handle to a result set on the server. A ref cursor is represented through the `OracleRefCursor` ODP.NET class.

Ref cursors have the following characteristics:

1. A ref cursor refers to server memory. The memory address represented by a ref cursor "lives" on the database server, not on the client machine. Therefore, the client's connection to the database must be in place during the lifetime of the ref cursor. If the underlying connection to the database is closed, the ref cursor will become inaccessible to the client.
2. A ref cursor involves an additional database round trip. Because a ref cursor is a pointer to memory on the server that is returned to the client, the actual data contained in the ref cursor is not initially returned to the client. The client must request the data contained in the ref cursor after it has opened the ref cursor. Note that data will not be retrieved until the user attempts to read it.
3. A ref cursor is not updatable. The result set represented by the ref cursor is read-only. You cannot update the database by using a ref cursor.
4. A ref cursor is not backward scrollable. The data represented by the ref cursor is accessed in a forward-only, serial manner. You cannot position a record pointer inside the ref cursor to point to random records in the result set.
5. A ref cursor is a PL/SQL datatype. You create and return a ref cursor inside a PL/SQL code block.

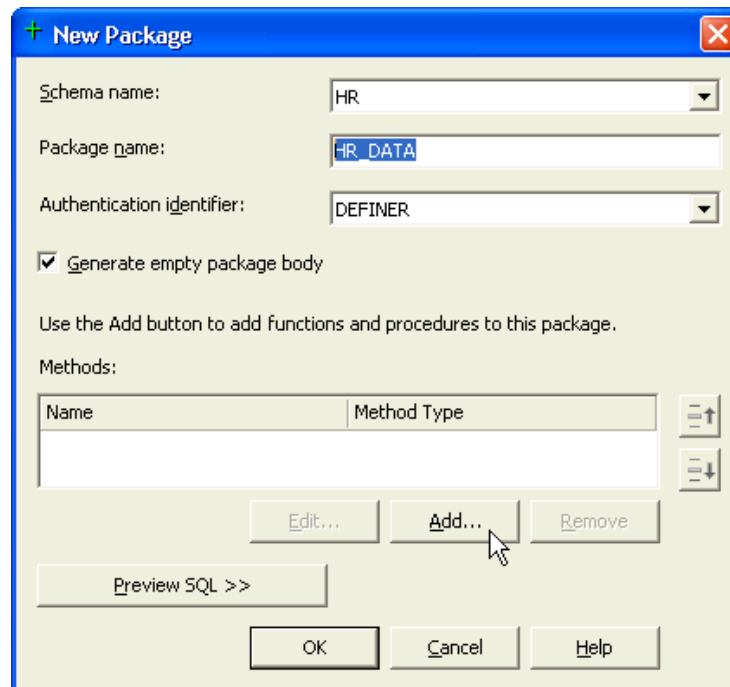
## Creating a PL/SQL Stored Procedure that Uses Ref Cursors

1. Follow the instructions in Section "Copying a Project" on page 4-1 to create a new copy of the `HR_DataSet_ODP_CS` project. Name the new project `HR_StoredProcedure_CS`. If using VB, name it `HR_StoredProcedure_VB`.
2. In Oracle Explorer, right-click **Packages** and select **New Package**.



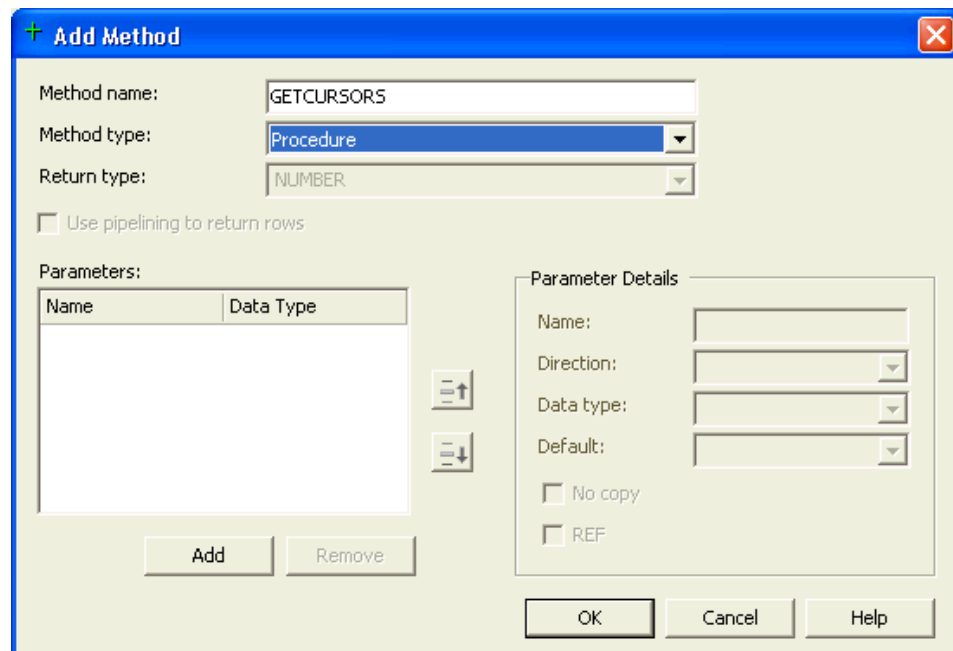
The New Package window appears.

3. In the New Package window, change the **Package Name** to `HR_DATA`.
4. Click **Add**.



The Add Method window appears.

5. In the Add Method window, enter **Method Name** GETCursors, and change **Method Type** to Procedure.



6. Under Parameters, click **Add**.

In the Add Method window, under Parameter Details, enter DEPARTMENTID for **Name**, IN for **Direction**, and NUMBER for **Data Type**.

**Add Method**

Method name: GETCursors

Method type: Procedure

Return type: NUMBER

☐ Use pipelining to return rows

Parameters:

Name	Data Type
DEPARTMENTID	NUMBER

Add Remove

Parameter Details

Name: DEPARTMENTID

Direction: IN

Data type: NUMBER

Default:

☐ No copy

☐ REF

OK Cancel Help

7. Under Parameters, click **Add**.

Enter a second parameter under Parameter Details, with **EMPLOYEEESCUR** for **Name**, **OUT** for **Direction**, and **SYS\_REFCURSOR** for **Data Type**.

8. Under Parameters, click **Add**.

Enter a third parameter under Parameter Details, with **DEPENDENTSCUR** for **Name**, **OUT** for **Direction**, and **SYS\_REFCURSOR** for **Data Type**.

9. Click **OK**.

The New Package window appears.

10. In the New Package window, click **Preview SQL** to see the SQL code created.

A Preview SQL window appears, containing code in [Example 6-1](#). Note that this example has been abbreviated by removing most of the comments.

#### **Example 6-1 PL/SQL Code for Package HR\_DATA**

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

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

    PROCEDURE "GETCursors" (
        "DEPARTMENTID" IN NUMBER,
        "EMPLOYEEESCUR" OUT SYS_REFCURSOR,
        "DEPENDENTSCUR" 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" (
    "DEPARTMENTID" IN NUMBER,
    "EMPLOYEEESCUR" OUT SYS_REFCURSOR,
    "DEPENDENTSCUR" 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";

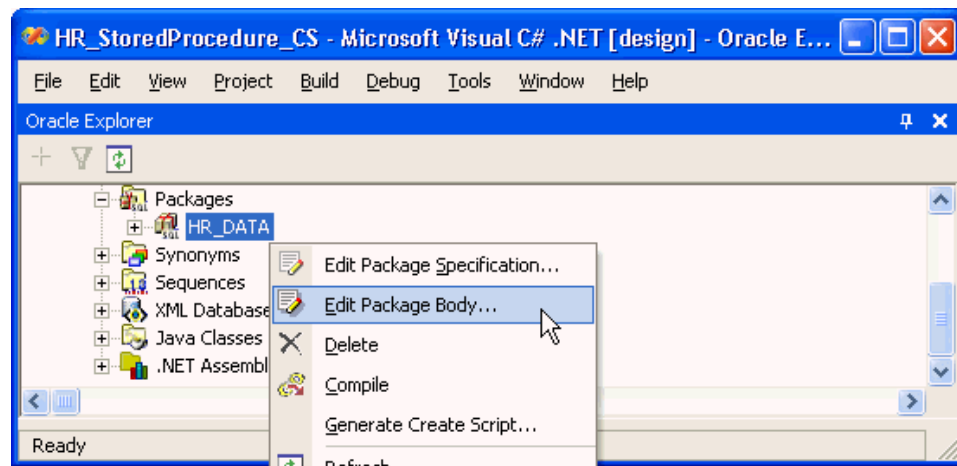
```

Click **OK** to close the window.

11. In the New Package window, click **OK**.

Note that a new package, HR\_DATA, now appears in Oracle Explorer.

12. In Oracle Explorer, right-click package HR\_DATA, and select **Edit Package Body**.



The code for the package is displayed.

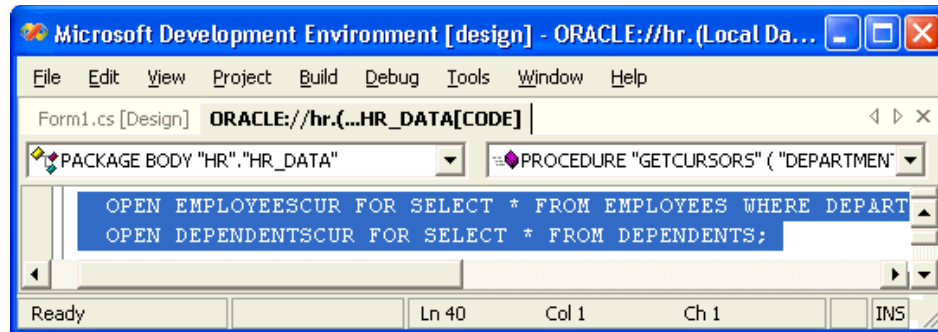
13. Scroll down to the body of the GETCURSORS procedure, and replace NULL; with code in [Example 6-2](#):

#### **Example 6-2 Assigning Reference Cursors**

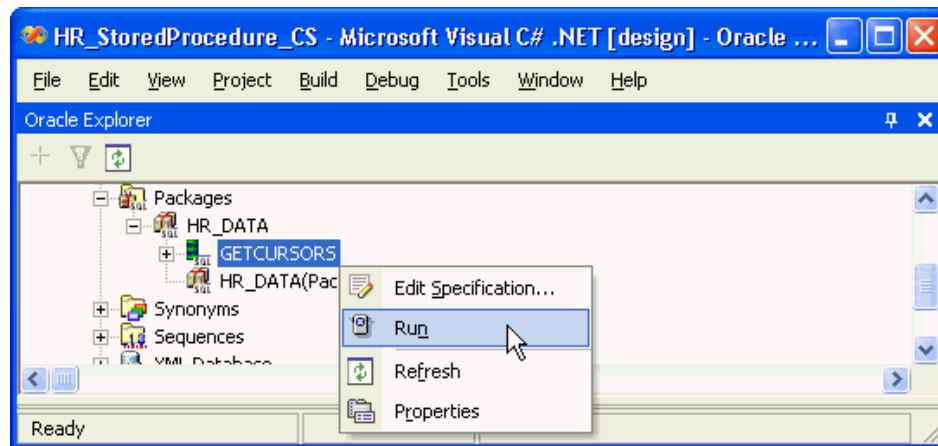
```

OPEN EMPLOYEEESCUR FOR SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID=DEPARTMENTID;
OPEN DEPENDENTSCUR FOR SELECT * FROM DEPENDENTS;

```

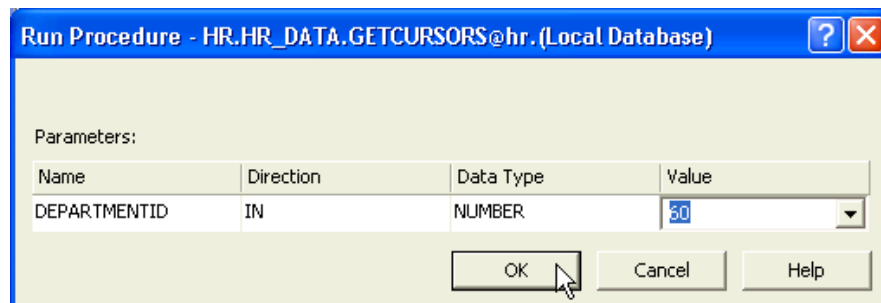


14. Save the changes to the package; use the **Ctrl+S** keyboard shortcut.
15. To run the stored procedure, in Oracle Explorer, expand package **HR\_DATA**. Right-click the **GETCURSORS** method, and select **Run**.



A Run Procedure window appears.

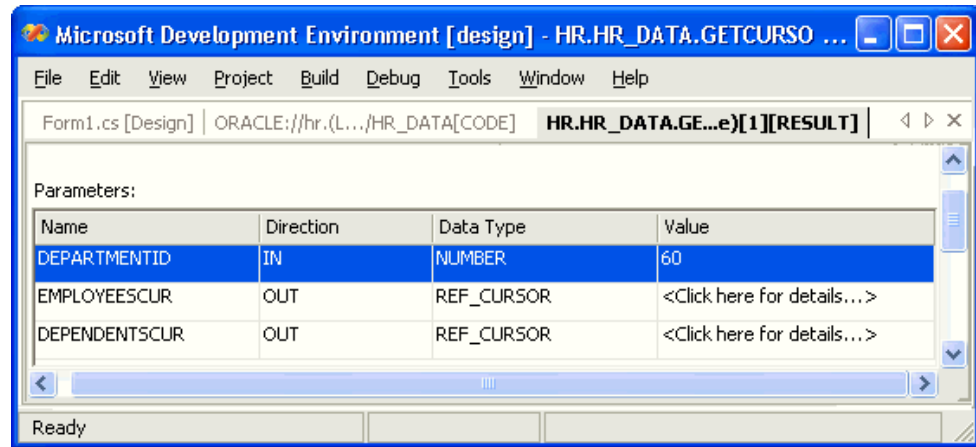
16. In the Run Procedure window, enter a **Value** of 60 for **DEPARTMENTID**. Click **OK**.



17. The Output window appears, showing that the run was successful. Close the Output Window.
18. In the design view, the following message appears:

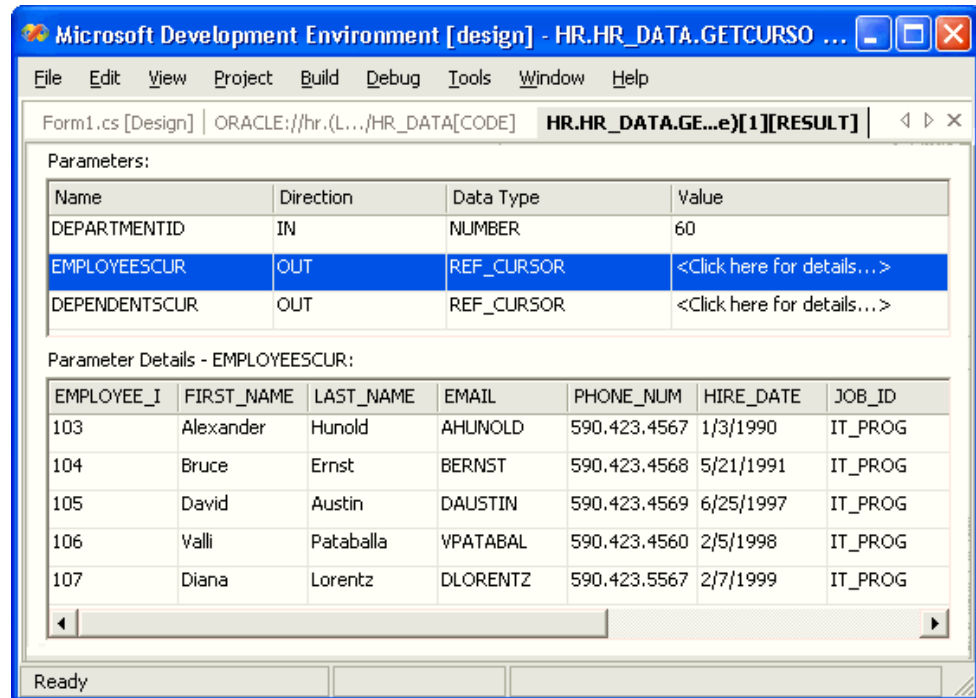
Procedure <HR.HR\_DATA.GETCURSORS@hr.database> was run successfully.

Under this message, note two new parameters (together with **DEPARTMENTID**): **EMPLOYEE\_SCUR** and **DEPENDENT\_SCUR**.



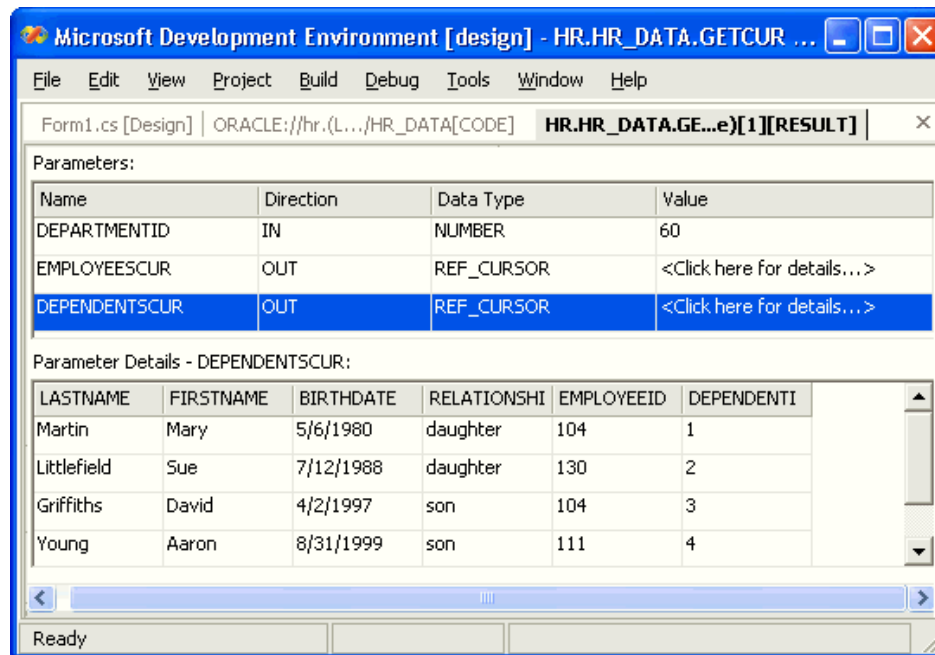
19. In the design view, select the **Value** column entry for EMPLOYEEESCUR.

The Parameter Details area appears, showing the result of the EMPLOYEEESCUR for DEPARTMENTID 60.



20. In the design view, select the **Value** column entry for DEPENDENTSCUR.

The Parameter Details area appears, showing the result of the DEPENDENTSCUR.



## Running a PL/SQL Stored Procedure Using Oracle Data Provider for .NET

1. With Form1 selected, switch to code view using the F7 keyboard shortcut.
2. In the try block of the connect\_Click() method, replace the two command assignment lines, starting with cmd = New OracleCommand... with code in [Example 6-3](#) or [Example 6-4](#).

### **Example 6-3 Changing OracleCommand to Use a Stored Procedure: C#**

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

### **Example 6-4 Changing OracleCommand to Use a Stored Procedure: VB**

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

3. Under the code added in Step 2, add definitions and bindings for the three parameters of the GETCURSORS stored procedure as OracleParameter objects, calling them dept\_id, emp\_cur and dnt\_cur.

### **Example 6-5 Defining and Binding OracleParameter Objects for Stored Procedure: C#**

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

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

```

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

```

**Example 6-6 Defining and Binding OracleParameter Objects for Stored Procedure: VB**

```

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

```

```

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

```

```

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

```

4. Run the application using the F7 keyboard shortcut.

A Form1 window appears.

5. In the Form1 window, enter the connection information, and click **Connect**.
6. In the DataGrid object, scroll horizontally to note that the last column, DEPARTMENT\_ID, is equal to 60.

Note that the DataGrid contains the first result set from the stored procedure, which matches the query of the EMPLOYEES table.

	EMPLOYEE_	FIRST_NAM	LAST_NAME	EMAIL	PHONE_NU	HIRE_C
▶	103	Alexander	Hunold	AHUNOLD	590.423.4567	1/3/199
	104	Bruce	Ernst	BERNST	590.423.4568	5/21/19
	105	David	Austin	DAUSTIN	590.423.4569	6/25/19
	106	Valli	Pataballa	VPATABAL	590.423.4560	2/5/199
	107	Diana	Lorentz	DLORENTZ	590.423.5567	2/7/199

7. Close the application.



---

## Deploying .NET Stored Procedures

This chapter discusses how to use and deploy .NET stored procedures in your application. You can use custom stored procedures in your ODP.NET code in the same manner as any other stored procedure.

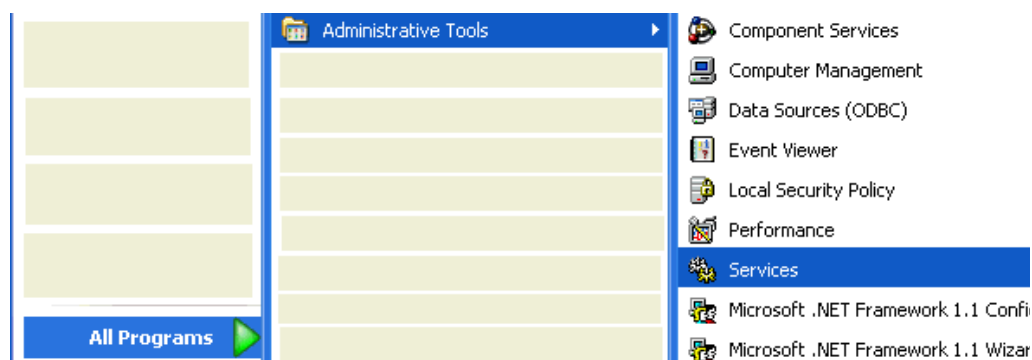
This chapter contains the following sections:

- [Starting the Common Language Runtime Service](#)
- [Creating an Oracle Project](#)
- [Creating a New Connection](#)
- [Creating .NET Stored Functions and Procedures](#)
- [Deploying .NET Stored Functions and Procedures](#)
- [Running .NET Stored Functions and Procedures](#)

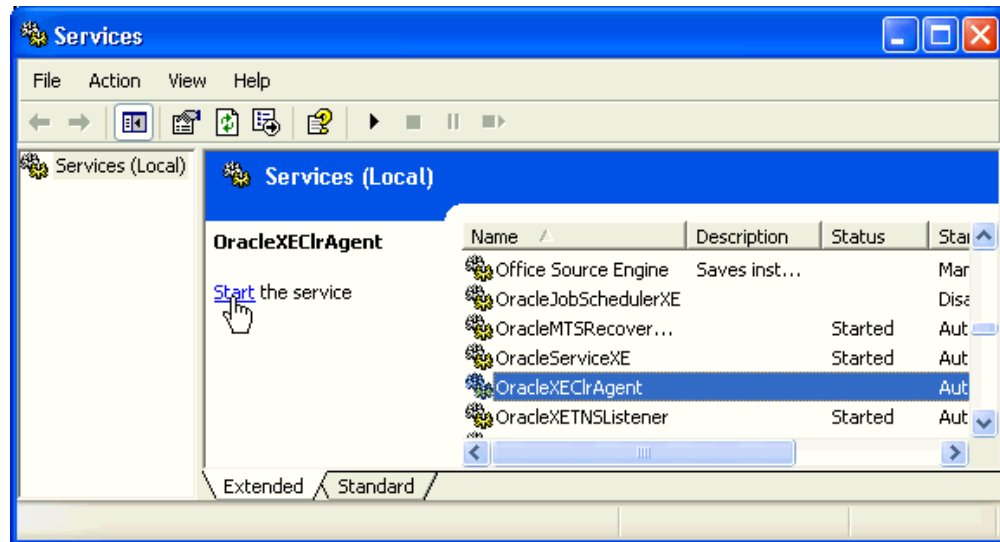
### Starting the Common Language Runtime Service

To use .NET Stored Procedures, you must first start the XE Common Language Runtime agent, represented by the `OracleXEClrAgent` service. This service may not start by default. Note that it is located on the Oracle XE database server, not on the client.

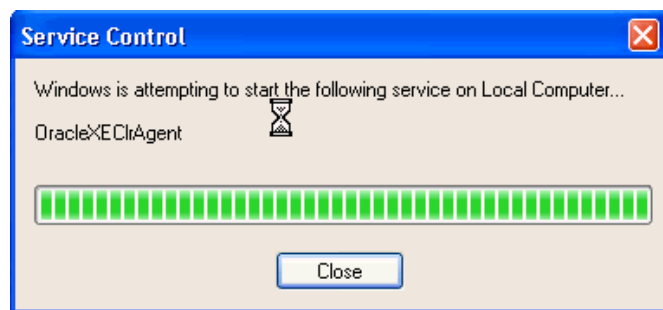
1. From the **Start** menu, select **All Programs**, then select **Administrative Tools**, and finally, select **Services**.



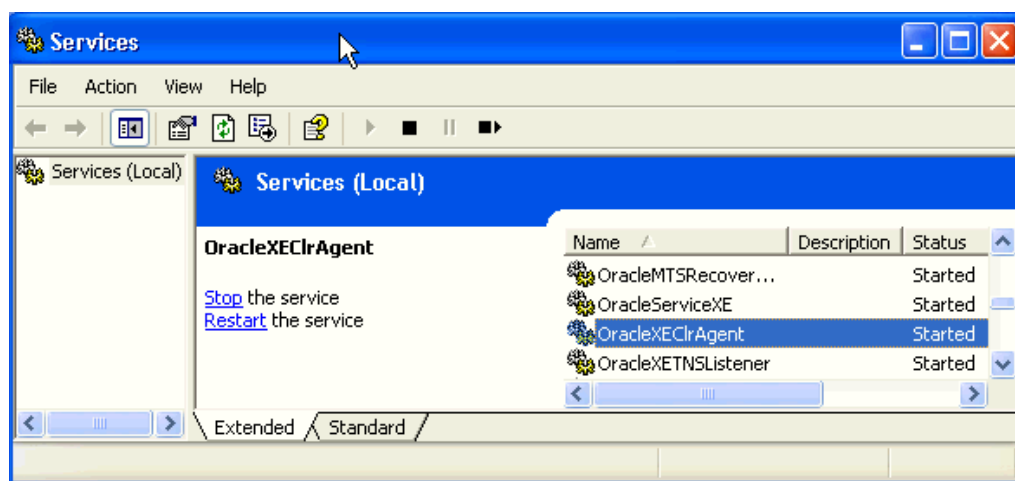
2. In the Services window, click the **Extended** tab.  
Scroll down the list of Services, and select **OracleXEClrAgent**.  
Click the **Start** hyperlink.



3. The Service Control window shows that the OracleXEClrAgent is starting.



4. When the Service Control window closes, note that the status of the OracleXEClrAgent is changed to Started.



## Creating an Oracle Project

Follow these steps to start a new Oracle project in Visual Studio .NET 2003:



1. From the **File** menu, select **New**, and then select **Project**.

A New Project dialog box appears.

2. On the left side of the New Project dialog box under Project Types, select **Visual C# Projects**.

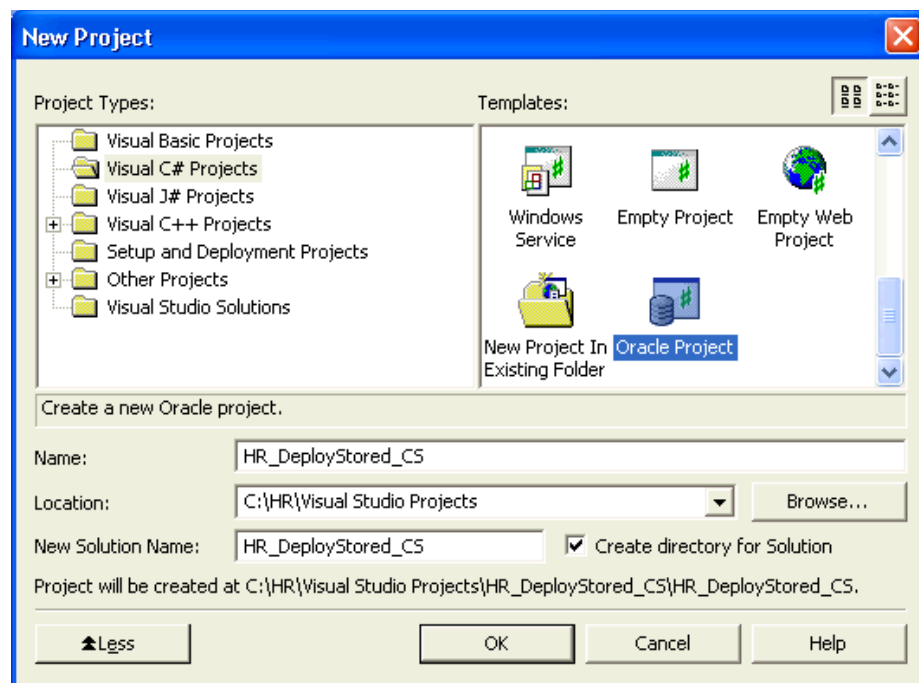
On the right side of the New Project dialog box under Templates, select **Oracle Project**.

For **Name**, enter HR\_DeployStored\_CS.

For **Location**, enter C:\HR\Visual Studio Projects.

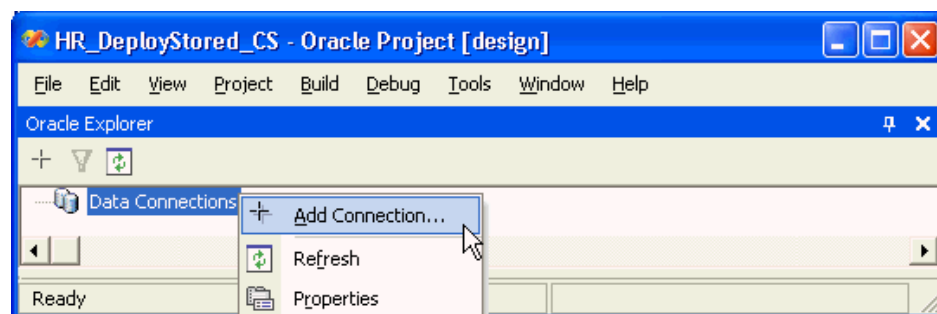
Click **OK**.

If you wish to create project in Visual Basic, under Project Types select **Visual Basic Projects** instead, and enter HR\_DeployStored\_VB under **Name**.



## Creating a New Connection

1. In Oracle Explorer, right-click **Data Connections**. From the menu, select **Add Connection**.



An Add Connection window appears.

2. In the Add Connection window, enter the following information:

**Data source name:** Use the `Local Database` if you are connecting to a database on the same machine. Otherwise, use the alias of the remote database instance.

Select the **Use a specific user name and password** option.

For **User name**, enter `hr`.

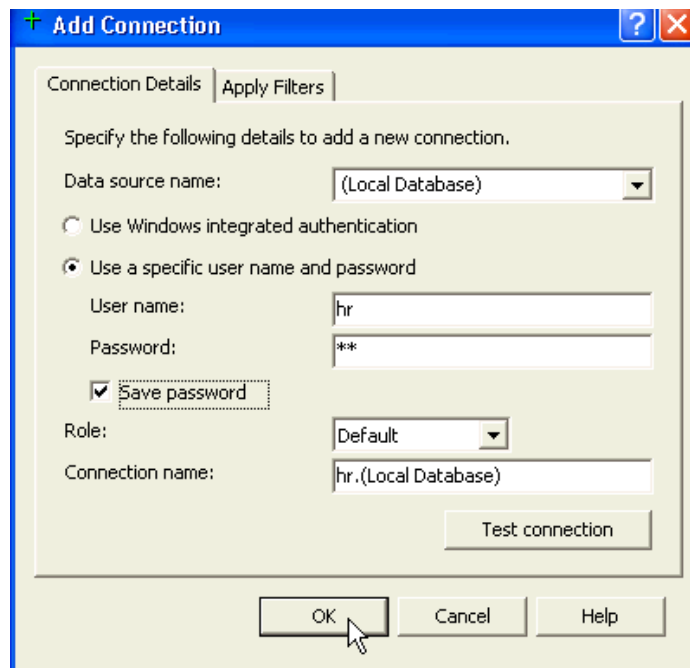
For **Password**, enter `hr`, or the password that was set when unlocking and setting up the `hr` account.

To save the password for future sessions, check the **Save password** box. While it is not advisable to save connection password within your application in clear text for security reasons, we will do it in this demonstration.

Ensure that the **Role** is set to `Default`. This refers to the default roles that have been granted to the user `hr`.

The **Connection name** should be generated automatically from the **Data source name** and the **User name** values.

Click the **Apply Filters** tab and check that the `HR` schema is in the Displayed schemas column. Only the schema objects (tables, views, and so on) from the schemas selected in the Apply Filters tab are displayed when you expand the schema category nodes in the data connection.



Click the **Connection Details** tab, and click **Test connection**.



The test should succeed. Click **OK**.

If the test fails, it may be due to one or more of the following issues that you must address before proceeding with further steps:

- The database is not started.
  - The database connectivity is not properly configured.
  - You do not have the correct user name, password, and role.
3. Click **OK** to close the Add Connection window.

The **Oracle Explorer** window should now contain the hr . (Local Database) connection.

## Creating .NET Stored Functions and Procedures

1. Select **Class1.cs** tab in your project.
2. Paste the `getDepartmentno()` method into the `Class1` declaration, as shown in [Example 7-1](#) and [Example 7-2](#).

### **Example 7-1 Adding `getDepartmentno()` Method Code: 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;
}
```

**Example 7-2 Adding getDepartmentno() Method Code: VB**

```

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()

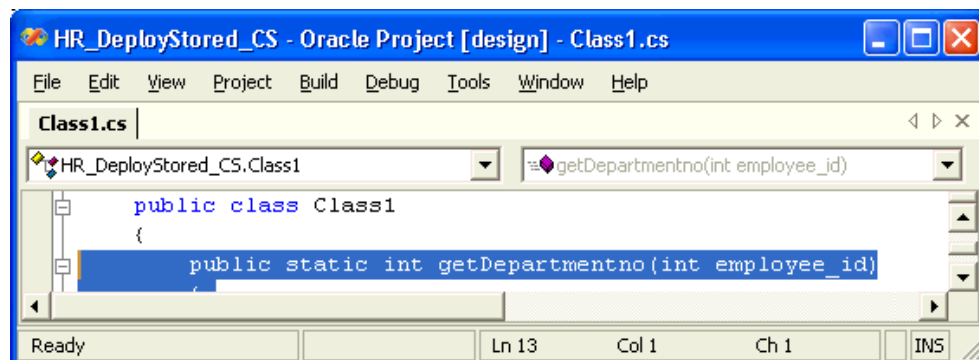
    ' Create and execute a command
    Dim cmd As OracleCommand = conn.CreateCommand()
    cmd.CommandText = "select department_id from employees where employee_id = :1"
    cmd.Parameters.Add(":1", OracleDbType.Int32, employee_id,
        ParameterDirection.Input)
    Dim rdr As OracleDataReader = cmd.ExecuteReader()

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

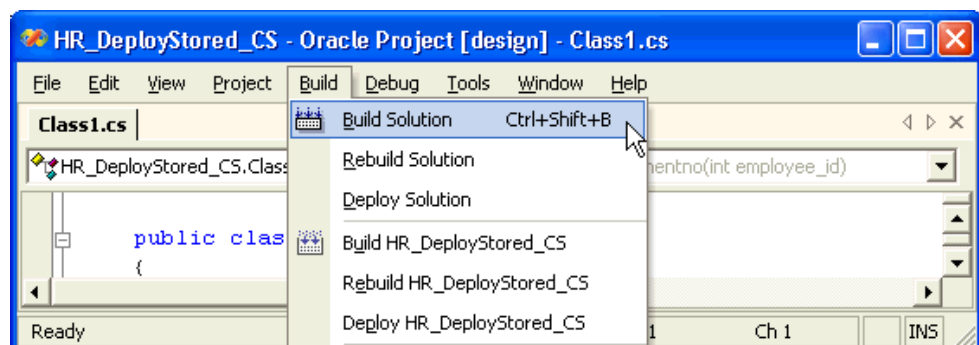
    rdr.Close()
    cmd.Dispose()

    ' Return the employee's department number
    getDepartmentno = department_id
End Function

```



3. Using the **Ctrl+S** keyboard shortcut, save Class1.
4. From the **Build** menu, select **Build Solution**.

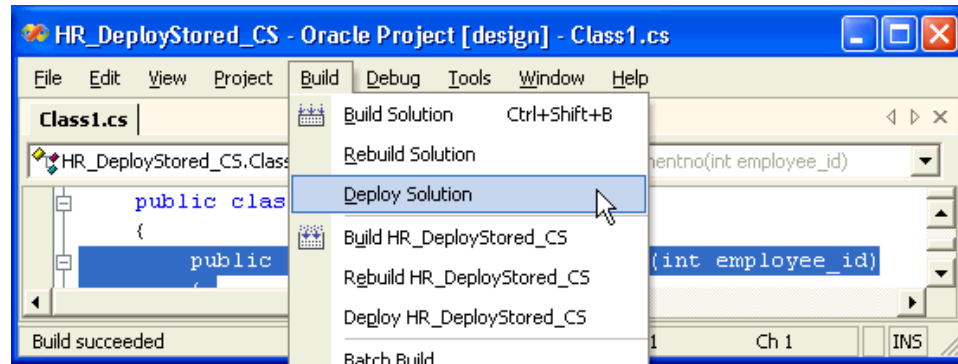


5. An Output window shows that the build was successful. Close the Output window.

## Deploying .NET Stored Functions and Procedures

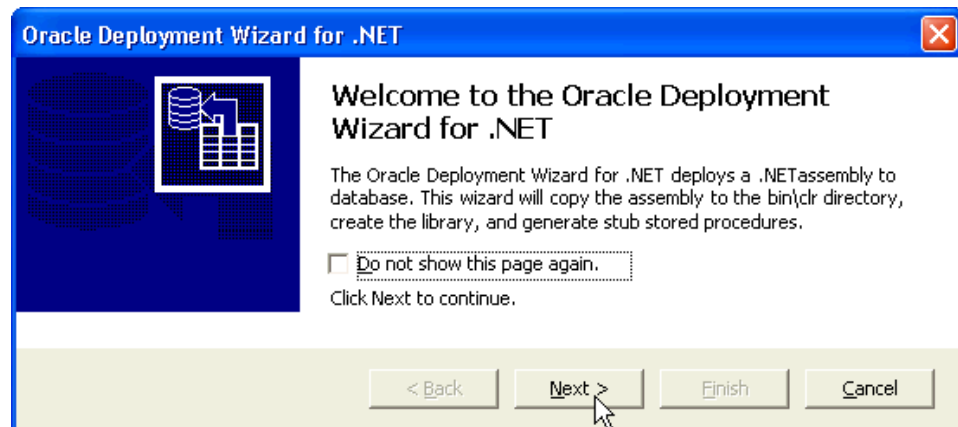
Follow these steps to deploy a .NET Stored Procedure:

1. From the **Build** menu, select **Deploy Solution**.



An Oracle Deployment Wizard for .NET window appears.

2. In the Oracle Deployment Wizard for .NET window, click **Next**.

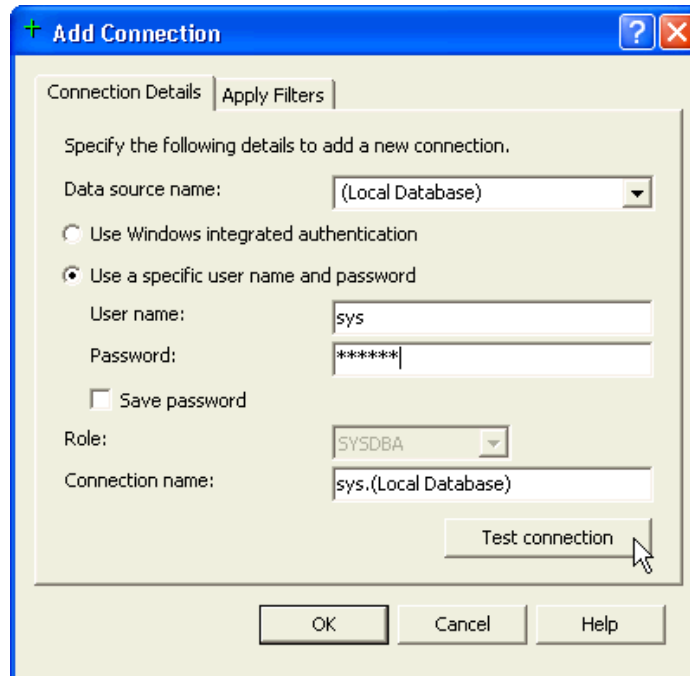


3. On the **Configure Your Connection** window, click **New Connection**.



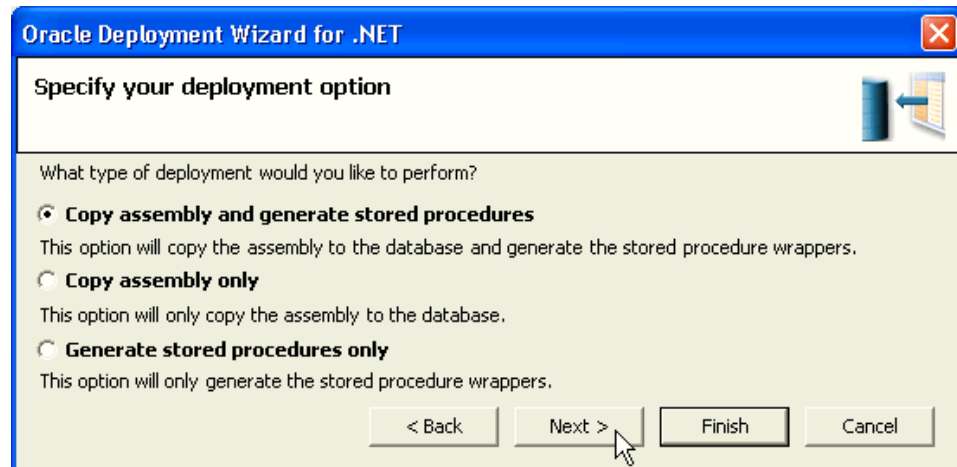
4. You must now establish a connection with SYSDBA privileges.
  - For the **Data Source Name**, use (Local Database).
  - For **Username**, enter sys.
  - For **Password**, enter the current sys password.
  - Click **Test Connection**.

To use the Oracle Application Express to set the sys account password, see Chapter 6, "Managing Users and Security" in the *Oracle Database Express Edition 2 Day DBA*.

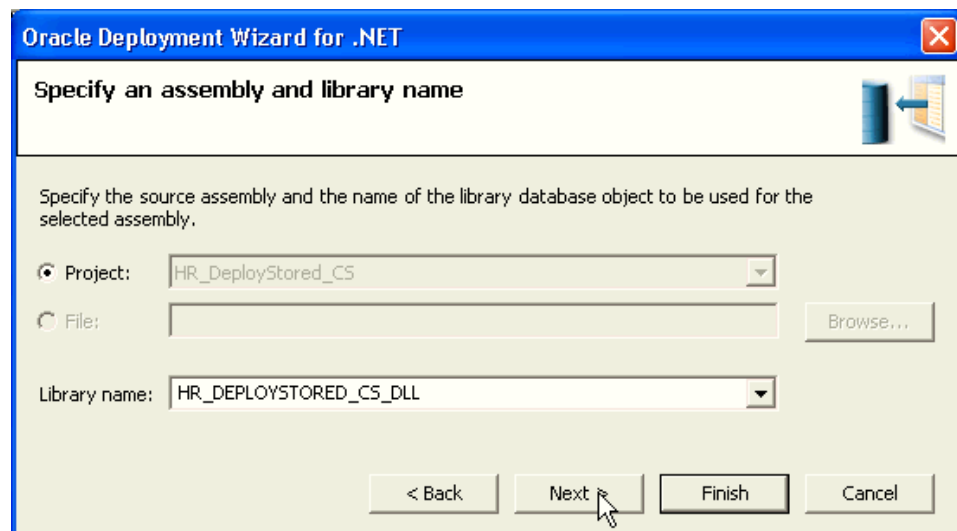


The test result window appears.

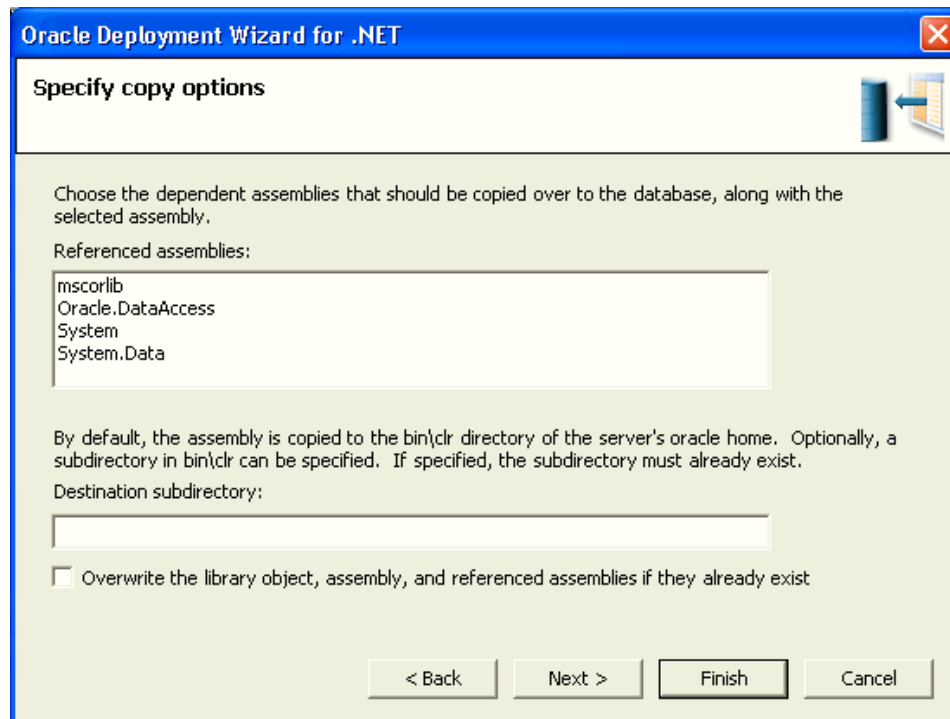
5. Click **OK** to close the test result window.
6. In the Add Connection window, click **OK**.
7. In the Oracle Deployment Wizard for .NET window, click **Next**.
8. On the Specify your deployment option window, ensure that **Copy assembly and generate stored procedures** is selected, and click **Next**.



9. On the Specify an assembly and library name window, accept the defaults and click Next.



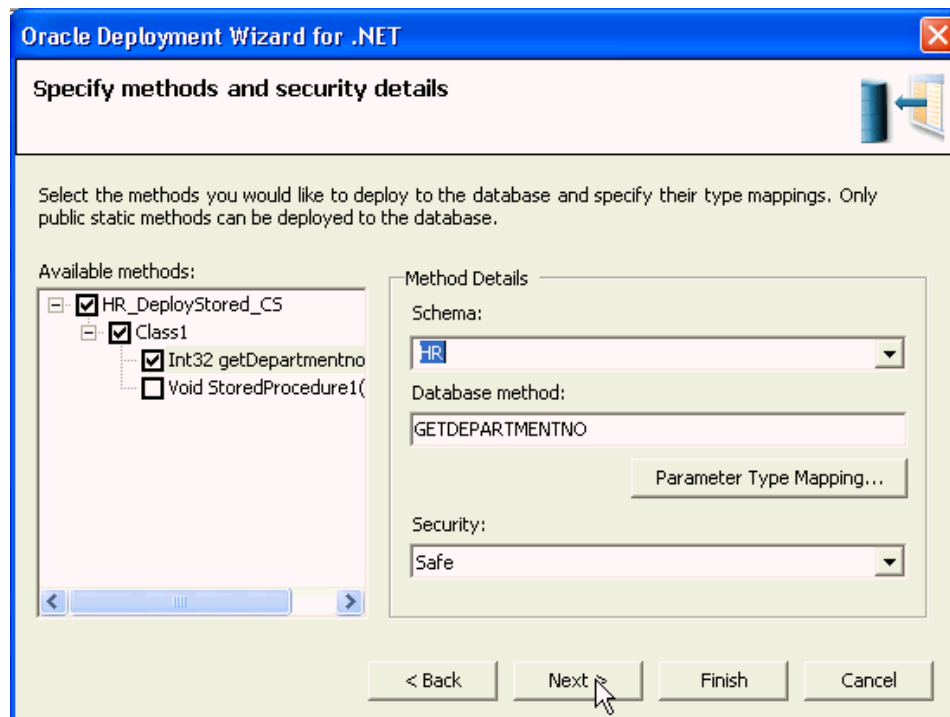
10. On the Specify copy options window, accept the defaults and click Next.



11. On the Specify methods and security details window, under Available methods, expand **HR\_DeployStored\_CS** (or **HR\_DeployStored\_VB**), then expand **Class1**, and select the **getDepartmentno()** method.

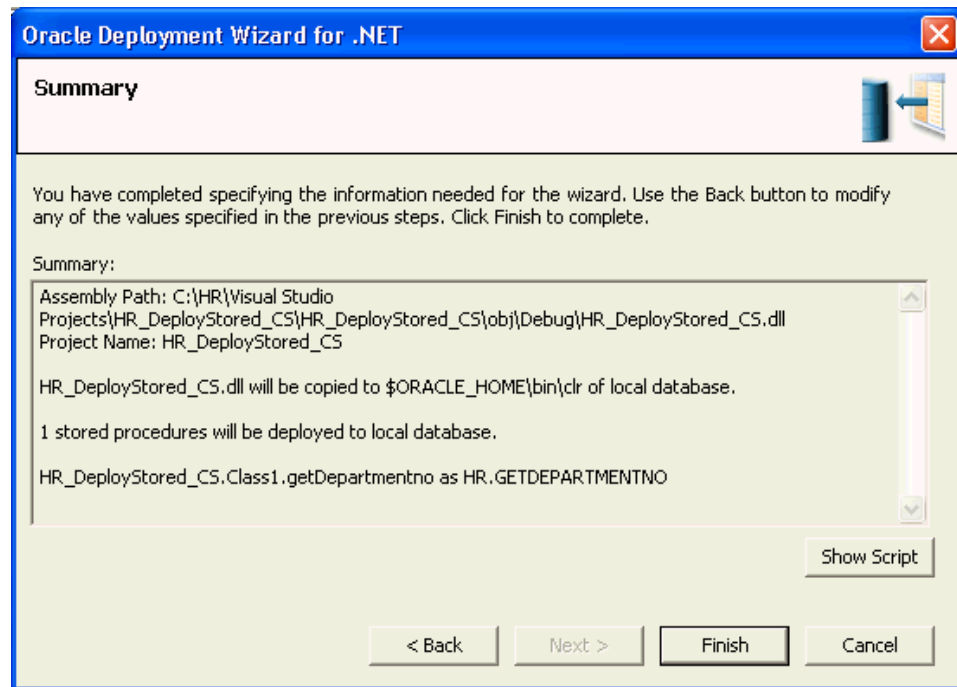
Under Method Details, select **HR** from the Schema drop-down list.

Click **Next**.





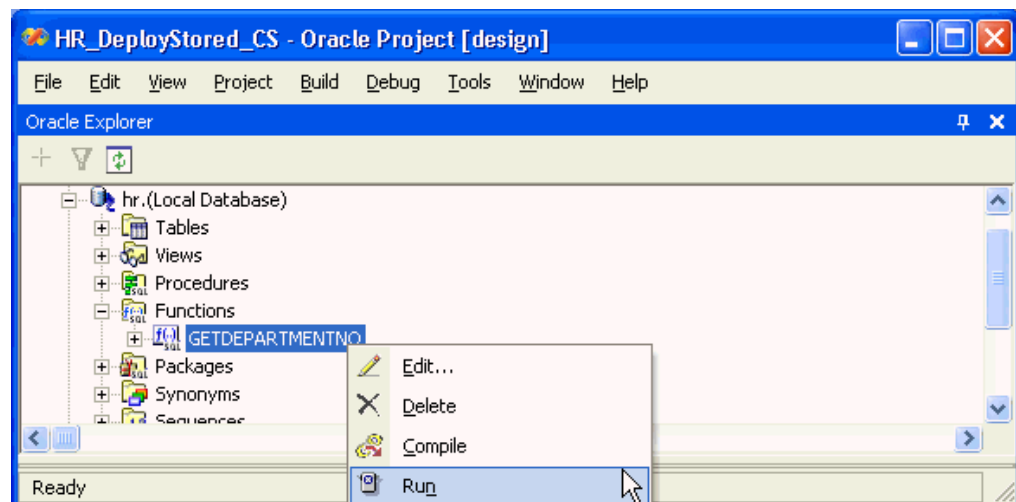
12. On the Summary window, click **Finish**.



## Running .NET Stored Functions and Procedures

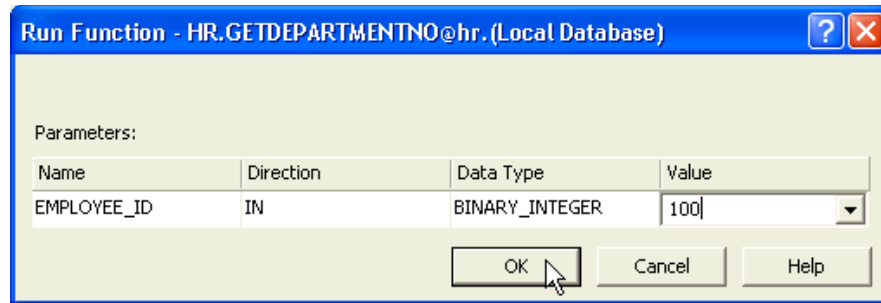
Follow these steps to run the .NET Stored procedure you created in Section ["Deploying .NET Stored Functions and Procedures"](#):

1. In Oracle Explorer, expand the hr . (LocalDatabase) connection. Expand Functions. Right-click GETDEPARTMENTNO and select **Run**.

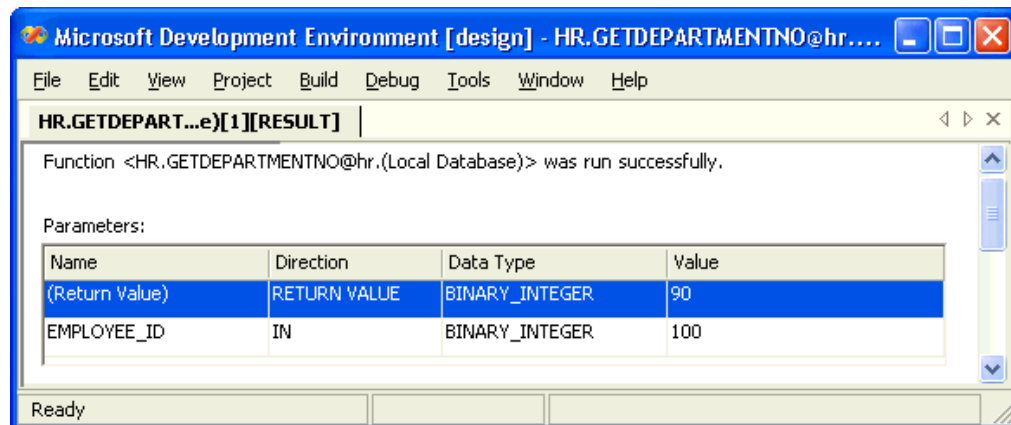


The Run Function window appears.

2. In the Run Function window, enter a Value of 100 for `EMPLOYEE_ID`. Click **OK**.



3. Note that the return value for department is 90.



---

## Including Globalization Support

This chapter discusses global application development with Oracle Database XE in .NET. It addresses the basic tasks associated with developing applications that are ready for global deployment, such as developing locale awareness and presenting data with cultural conventions of the user's locale. It also discusses globalization support features available in Oracle Data Provider for .NET.

This chapter contains the following sections:

- [Introduction to Global Applications](#)
- [Developing Global Applications with the .NET Framework](#)
- [Presenting Data in the Correct User Local Convention](#)
- [Synchronizing the .NET and Oracle Database Locale Environments](#)
- [Client Globalization Support in Oracle Data Provider for .NET](#)

### See Also:

- Chapter 8, "Oracle Data Provider for .NET Globalization Classes" in *Oracle Data Provider for .NET Developer's Guide*
- Chapter 8, "Working in a Global Environment" in the *Oracle Database Express Edition 2 Day Developer Guide*
- Microsoft .NET Internationalization Internet site, <http://www.microsoft.com/globaldev/getwr/>

## Introduction to Global Applications

Building a global-ready application that supports different locales requires good development practices. A locale refers to a national language and the region in which the language is spoken. The application itself must be aware of the user's locale preference and be able to present content following the cultural convention expected by the user. It is important to present data with appropriate locale characteristics, such as using the correct date and number formats. Oracle Database Express is fully internationalized to provide a global platform for developing and deploying global applications.

## Developing Global Applications with the .NET Framework

When planning a global-ready application, you have to consider two main tasks:

- **Globalization** is the process of designing applications that can adapt to different cultures.

- **Localization** is the process of translating resources for a specific culture.

In the .NET Framework, the `System.Globalization` namespace contains classes that define information related to culture, such as language, country and region, calendars, format patterns for dates, currency, and numbers, and the sort order for strings. These classes simplify the process of developing a global-ready application, so that passing a `CultureInfo` object that represents the user's culture to methods in `System.Globalization` namespace initiates the correct set of rules and data.

The .NET Framework also supports the creation and localization of resources, and offers a model for packaging and deploying them. Localizing the application's resources for specific cultures supports development of translated versions of the application. The .NET Framework's base class library provides several classes in the `System.Resources` namespace for building and manipulating application resources.

## Presenting Data in the Correct User Local Convention

Data in the application must be presented in a way that meets the user's expectations, or its meaning can be misinterpreted. For example, '12/11/05' implies 'December 11, 2005' in the United States and 'November 12, 2005' in the United Kingdom. Similar confusion exists for number and monetary formats. For example, the period (.) is a decimal separator in the United States and a thousand separator throughout Europe.

Different languages have their own sorting rules: some languages are collated according to the letter sequence in the alphabet, others according to stroke count in the letter, still others are ordered by the pronunciation of the words. Presenting data that is not sorted according to the linguistic sequence to which the user is accustomed can make searching for information difficult and time-consuming.

Depending on the application logic and the volume of data retrieved from the database, it may be more appropriate to format the data at the database level rather than at the application level. Oracle Database XE offers many features that refine the presentation of data when the user locale preference is known.

## Oracle Date Formats

There are three different date presentation formats in Oracle Database XE: standard, short, and long. The following steps illustrate the difference between the short and long date formats for United States and Germany.

1. Using SQLPlus, connect to the database and enter the command in [Example 8–1](#) at the SQL prompt.

### ***Example 8–1 Setting NLS\_TERRITORY and NLS\_LANGUAGE Parameters: United States***

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

This message appears: Session altered.

2. At the SQL prompt, enter the query in [Example 8–2](#).

### ***Example 8–2 Testing the NLS Date Format Settings***

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

- The result of the query returns in the American format specified in Step 1.

ID	Name	Short Hire	Long Hire	Date
100	S. King	6/17/1987	Wednesday, June 17, 1987	
101	N. Kochhar	9/21/1989	Thursday, September 21, 1989	
102	L. De Haan	1/13/1993	Wednesday, January 13, 1993	
103	A. Hunold	1/3/1990	Wednesday, January 03, 1990	
104	B. Ernst	5/21/1991	Tuesday, May 21, 1991	

- Enter the command in [Example 8-3](#) at the SQL prompt.

**Example 8-3 Setting NLS\_TERRITORY and NLS\_LANGUAGE Parameters: Germany**

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

This message appears: Session wurde geändert.

- At the SQL prompt, enter the query in [Example 8-2](#).
- The result of the query returns in the German format specified in Step 4.

ID	Name	Short Hire	Long Hire	Date
100	S. King	17.06.1987	Mittwoch, 17. Juni 1987	
101	N. Kochhar	21.09.1989	Donnerstag, 21. September 1989	
102	L. De Haan	13.01.1993	Mittwoch, 13. Januar 1993	
103	A. Hunold	03.01.1990	Mittwoch, 3. Januar 1990	
104	B. Ernst	21.05.1991	Dienstag, 21. Mai 1991	

## Oracle Number Formats

There are also differences in the decimal character and group separator. The following steps illustrate these difference between United States and Germany.

- Enter the command in [Example 8-4](#) at the SQL prompt.

**Example 8-4 Setting NLS\_TERRITORY Parameter: United States**

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

This message appears: Session altered.

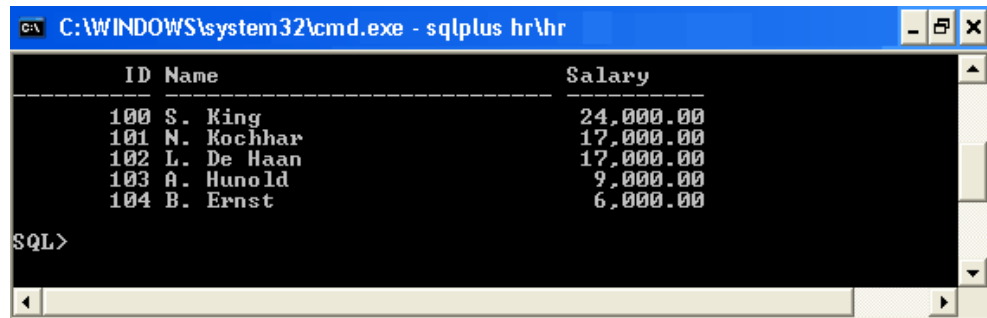
- At the SQL prompt, enter the query in [Example 8-5](#).

**Example 8-5 Testing NLS Number Format Settings**

```
SQL> SELECT employee_id "ID",
2 SUBSTR (first_name,1,1)||'. '||last_name "Name",
3 TO_CHAR (salary, '99G999D99') "Salary"
5 FROM employees
```

```
6 WHERE employee_id < 105;
```

3. The result of the query returns in the American format specified in Step 1.



ID	Name	Salary
100	S. King	24,000.00
101	N. Kochhar	17,000.00
102	L. De Haan	17,000.00
103	A. Hunold	9,000.00
104	B. Ernst	6,000.00

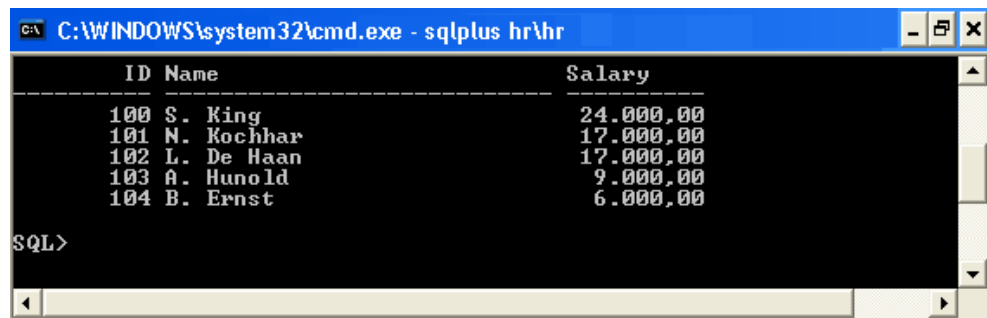
4. Enter the command in [Example 8-6](#) at the SQL prompt.

**Example 8-6 Setting NLS\_TERRITORY Parameter: Germany**

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

This message appears: Session altered.

5. At the SQL prompt, enter the query in [Example 8-5](#).
6. The result of the query returns in the German format specified in Step 4.



ID	Name	Salary
100	S. King	24.000,00
101	N. Kochhar	17.000,00
102	L. De Haan	17.000,00
103	A. Hunold	9.000,00
104	B. Ernst	6.000,00

## Oracle Linguistic Sorts

Spain traditionally treats *ch*, *ll*, and *ñ* as letters of their own, ordered after *c*, *l* and *n*, respectively. The following steps illustrate the effect of using a Spanish sort against the employee names Chen and Chung.

1. Enter the command in [Example 8-7](#) at the SQL prompt.

**Example 8-7 Setting NLS\_SORT Parameter: Binary**

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

This message appears: Session altered.

2. At the SQL prompt, enter the query in [Example 8-8](#).

**Example 8-8 Testing NLS Sort Order Settings**

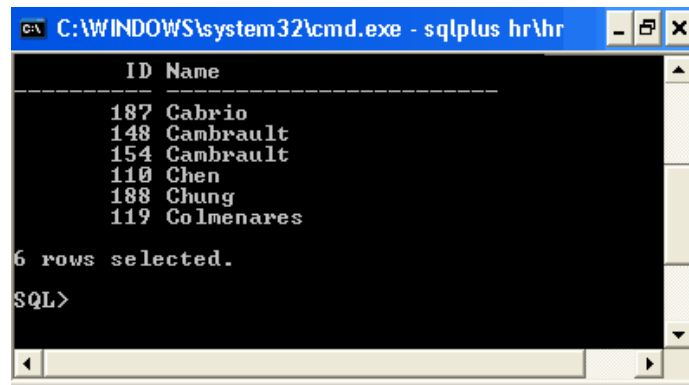
```
SQL> SELECT employee_id "ID",
2         last_name "Name",
```

```

3 FROM employees
4 WHERE employee_id < 105;

```

3. The result of the query returns in the binary sort specified in Step 1.



```

C:\WINDOWS\system32\cmd.exe - sqlplus hr\hr
-----
ID Name
-----
187 Cabrio
148 Cambrault
154 Cambrault
110 Chen
188 Chung
119 Colmenares

6 rows selected.
SQL>

```

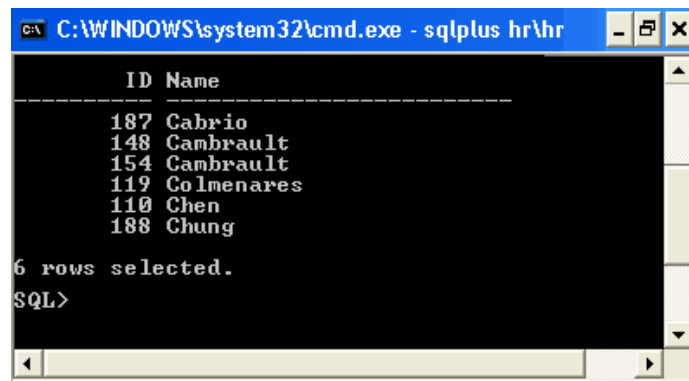
4. Enter the command in [Example 8-9](#) at the SQL prompt.

**Example 8-9 Setting NLS\_SORT Parameter: Spanish**

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

This message appears: Session altered.

5. At the SQL prompt, enter the query in [Example 8-8](#).
6. The result of the query returns in the Spanish sort specified in Step 4.



```

C:\WINDOWS\system32\cmd.exe - sqlplus hr\hr
-----
ID Name
-----
187 Cabrio
148 Cambrault
154 Cambrault
119 Colmenares
110 Chen
188 Chung

6 rows selected.
SQL>

```

## Oracle Error Messages

The NLS\_LANGUAGE parameter also controls the language of the database error messages. Setting this parameter prior to submitting a SQL query ensures the return of local language-specific error messages, as shown in these steps:

1. Enter the command in [Example 8-10](#) at the SQL prompt.

**Example 8-10 Setting NLS\_LANGUAGE Parameter: United States**

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

This message appears: Session altered.

- At the SQL prompt, enter the query in [Example 8–11](#).

**Example 8–11 Testing NLS Error Messages Settings**

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

- The result of the query return the error message in the language specified in Step 1.

```

C:\WINDOWS\system32\cmd.exe - sqlplus hr\hr
SQL> select * from managers;
select * from managers
*
ERROR at line 1:
ORA-00942: table or view does not exist
SQL>

```

- Enter the command in [Example 8–9](#) at the SQL prompt.

**Example 8–12 Setting NLS\_LANGUAGE Parameter: French**

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

This message appears: Session modifiée.

- At the SQL prompt, enter the query in [Example 8–11](#).
- The result of the query returns the error message in the language specified in Step 4.

```

C:\WINDOWS\system32\cmd.exe - sqlplus hr\hr
SQL> select * from managers;
select * from managers
*
ERREUR à la ligne 1 :
ORA-00942: Table ou vue inexistante
SQL>

```

## Synchronizing the .NET and Oracle Database Locale Environments

When you are developing global applications across different programming environments, ensure that the user locale settings are always synchronized. Otherwise, the application may present conflicting culture-sensitive information. For example, a .NET application must map the application user's Culture ID to the correct NLS\_LANGUAGE and NLS\_TERRITORY parameter values before performing SQL operations.

Table shows some of the more common locales, as defined in .NET and Oracle environments.

**Table 8–1 Common NLS\_LANGUAGE and NLS\_TERRITORY Parameters**

Culture	Culture ID	NLS_LANGUAGE	NLS_TERRITORY
Chinese (P.R.C.)	zh-CN	SIMPLIFIED CHINESE	CHINA



**Table 8–1 (Cont.) Common NLS\_LANGUAGE and NLS\_TERRITORY Parameters**

Culture	Culture ID	NLS_LANGUAGE	NLS_TERRITORY
Chinese (Taiwan)	zh-TW	TRADITIONAL CHINESE	TAIWAN
English (U.S.A.)	en-US	AMERICAN	AMERICA
English (U.K.)	en-GB	ENGLISH	UNITED KINGDOM
French (Canada)	fr-CA	CANADIAN FRENCH	CANADA
French (France)	fr-FR	FRENCH	FRANCE
German	de	GERMAN	GERMANY
Italian	it	ITALIAN	ITALY
Japanese	ja	JAPANESE	JAPAN
Korean	ko	KOREAN	KOREA
Portuguese (Brazil)	pt-BR	BRAZILIAN PORTUGUESE	BRAZIL
Portuguese	pt	PORTUGUESE	PORTUGAL
Spanish	es	SPANISH	SPAIN

## Client Globalization Support in Oracle Data Provider for .NET

Oracle Data Provider for .NET enables applications to manipulate culture-sensitive data, such as ensuring proper string format, date, time, monetary, numeric, sort order, and calendar support using culture conventions defined in the Oracle Database XE. The default globalization settings are determined by the client's NLS\_LANG parameter, which is defined in the Windows Registry of the local computer. When the `OracleConnection` call `Open()` establishes a connection, it implicitly opens a session with globalization parameters specified by the value of the NLS\_LANG parameter.

### Client Globalization Settings

Client globalization settings derive from the Oracle globalization setting, NLS\_LANG, in the Windows Registry of the local computer. The client globalization parameter settings are read-only and remain constant throughout the lifetime of the application.

[Example 8–13](#) and [Example 8–14](#) illustrate how these settings can be obtained by calling the `OracleGlobalization.GetClientInfo()` static method. The properties of the `OracleGlobalization` object provide the Oracle globalization value settings.

#### **Example 8–13 How to Obtain Oracle Globalization Settings: 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 charsetset: " + ClientGlob.ClientCharacterSet);
    }
}
```

**Example 8–14 How to Obtain Oracle Globalization Settings: VB**

```
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 charsetset: " + ClientGlob.ClientCharacterSet)
    End Sub
End Class
```

## Session Globalization Settings

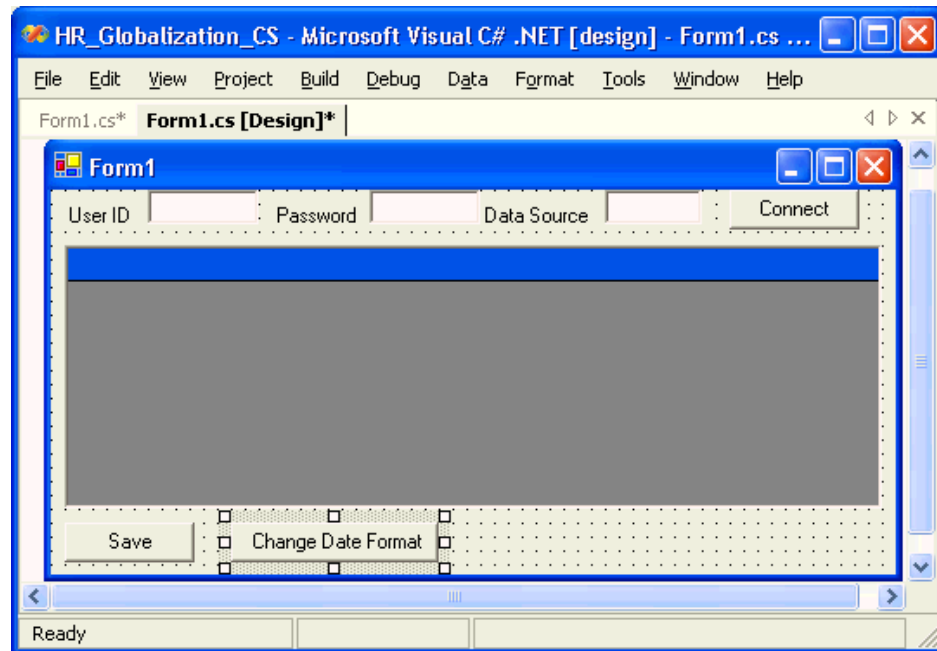
Session globalization parameters are initially identical to client globalization settings, but they can be updated. To obtain session globalization settings, first establish a connection to the database, and then call the `GetSessionInfo()` method of an `OracleConnection` object. The properties of the resulting `OracleGlobalization` object represent the globalization settings of the session.

When the `OracleConnection` object establishes a connection, it implicitly opens a session in which globalization parameters are initialized with values specified by the client's Oracle Globalization (or NLS) Registry settings. The globalization settings of a session can change during its lifetime.

To change globalization session settings programmatically, follow these steps:

1. Follow the steps in Section "Copying a Project" on page 4-1 to create a new copy of the `HR_DataSet_ODP_CS` (or `HR_DataSet_ODP_VB`) project. Name the new project `HR_Globalization_CS` (or `HR_Globalization_VB`).
2. Open **Form1** of the `HR_Globalization_CS` (or `HR_Globalization_VB`) project, and switch to design view (use the **Shift+F7** keyboard shortcut).
3. From the **View** menu, select **Toolbox**.
4. From the Toolbox, under Windows Forms, drag and drop a **Button** onto **Form1**.
5. Right-click the new **Button**, select **Properties**. A Properties window appears.
6. In the Properties window, set these properties:
  - Under Appearance, change **Text** to `Change Date Format`.
  - Under Design, change **(Name)** to `date`.
  - Click the lightning icon (events), and then click the highlighted **Click** event. From the drop-down window, select `date_Click`.

Close the **Properties** window.



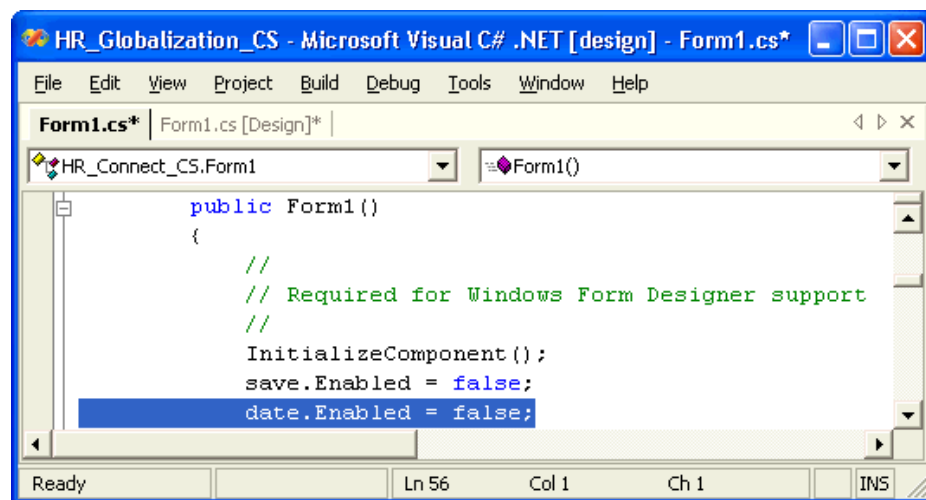
7. Switch to code view using the F7 keyboard shortcut.
8. Within the `Form1()` method, add the code shown in [Example 8-15](#) and [Example 8-16](#).

**Example 8-15 Disabling the Change Button: C#**

```
date.Enabled = false;
```

**Example 8-16 Disabling the Change Button: VB**

```
date.Enabled = false
```



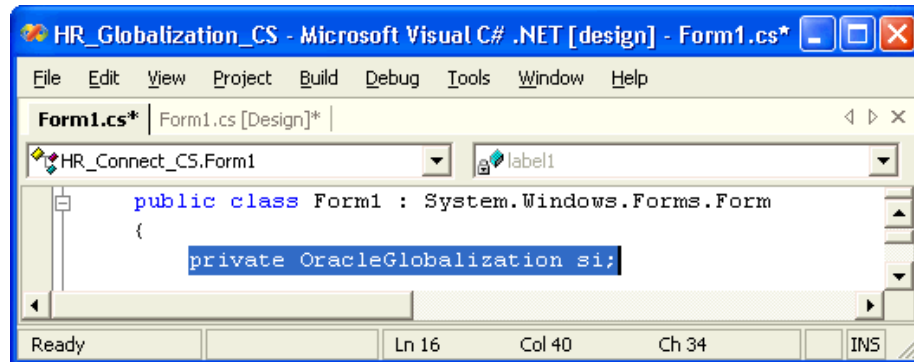
9. In the `Form1` class declarations, add the code shown in [Example 8-17](#) or [Example 8-18](#).

**Example 8-17 Creating an OracleGlobalization Object: C#**

```
private OracleGlobalization si;
```

**Example 8-18 Creating an OracleGlobalization Object: VB**

```
private si As OracleGlobalization
```



10. Within the `connect_Click()` method try block, as shown in [Example 8-19](#) and [Example 8-20](#),

- retrieve the value of the `OracleGlobalization` object
- retrieve data from the `EMPLOYEES` table (note the new query)
- enable the **Change Date Format** button

The changed code is in bold typeface.

**Example 8-19 Retrieving the Globalization Session Information: C#**

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

si = conn.GetSessionInfo();

string sql = "select employee_id, first_name, last_name, TO_CHAR(hire_date)
             from employees where employee_id < 105";
OracleCommand 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.Enabled = true;
```

**Example 8-20 Retrieving the Globalization Session Information: VB**

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

string sql = "select employee_id, first_name, last_name, TO_CHAR(hire date)
             from employees where employee_id < 105"
```

```

OracleCommand 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.Enabled = true

```

11. Your code also contains a new `date_Click()` method created in Step 6. Add to the method the code for changing the date format from the standard `DD-MON-RR` to `YYYY-MM-DD`, and for updating the `DataSet`, shown in [Example 8-21](#) and [Example 8-22](#). Note that the `ds.Clear()` call clears the old results before posting the changed data.

**Example 8-21 Changing the Date Format and Updating the DataSet: C#**

```

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

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

```

**Example 8-22 Changing the Date Format and Updating the DataSet: VB**

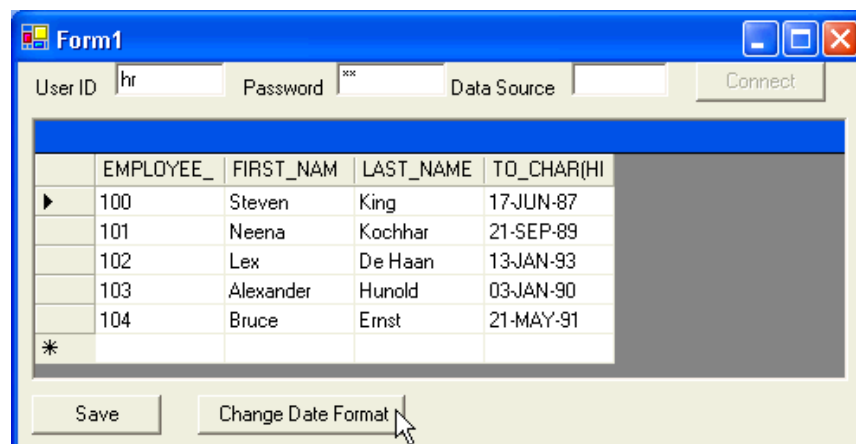
```

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

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

```

12. Save Form1 using **Ctrl+S** keyboard shortcut.
13. Run the application using the **F5** keyboard shortcut.
14. After you successfully connect to the database, the data grid is populated with the results of the query. Click **Change Date Format**.



15. Note that the date format is changed from the original DD-MON-RR to YYYY-MM-DD.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	TO_CHAR(HI)
100	Steven	King	1987-06-1
101	Neena	Kochhar	1989-09-2
102	Lex	De Haan	1993-01-1
103	Alexander	Hunold	1990-01-0
104	Bruce	Ernst	1991-05-2

16. Close the application.

## Thread-Based Globalization Settings

Thread-based globalization parameter settings are specific to each thread. Initially, these settings are identical to the client globalization parameters, but they can be changed programmatically. When converting ODP.NET Types to and from strings, use the thread-based globalization parameters, if applicable.

Thread-based globalization parameter settings are obtained by calling the `GetThreadInfo()` static method of the `OracleGlobalization` class. A call to `SetThreadInfo()` static method sets the thread's globalization settings.

ODP.NET classes and structures rely solely on the `OracleGlobalization` settings when manipulating culture-sensitive data. They do not use .NET thread culture information. If the application uses only .NET types, `OracleGlobalization` settings have no effect. However, when conversions are made between ODP.NET Types and .NET Types, `OracleGlobalization` settings are used where applicable.

---

**Note:** Changes to the `System.Threading.Thread.CurrentThread.CurrentCulture` property do not impact the `OracleGlobalization` settings of the thread or the session. The reverse is also true.

---

---

---

# Index

## A

---

account  
    unlocking, 5-2, 7-4  
Add Connection window, 5-1, 7-4, 7-8  
Add Method window, 6-3  
add reference, 3-3  
Add Reference dialog, 3-3  
Add() method, 4-6  
adding  
    reference, 3-2  
adding reference, 3-2  
alias  
    database, 5-2  
ALTER TABLE, 5-8  
application  
    result, 4-6  
    run, 3-14, 3-16, 3-17, 4-5, 4-8, 4-12, 5-12, 5-14, 6-9  
apply filters, 5-2, 7-4

## B

---

bind variable, 4-5  
    definition, 4-5  
    identifier, 4-6  
    name, 4-6  
    position, 4-6  
    update, 4-7  
BindByName property, 4-6  
binding data, 4-11  
build  
    deploy solution, 7-7  
    success, 7-7  
Build menu, 7-6, 7-7  
build solution, 7-6  
building connection, 3-13  
button, 3-10, 4-9  
    disable, 3-13, 3-14

## C

---

C  
    HRVisual Studio Projects directory, 4-1  
C# statement  
    using, 3-4  
Case statement, 3-17

class variable, 4-10  
Click event, 3-11  
click event, 4-10  
client globalization settings, 8-7  
close connection, 3-18  
CLR, 1-1  
Code view, 3-4  
code view, 5-12, 5-13, 6-8  
column  
    add, 5-4  
    data type, 5-4  
    name, 5-4  
    property, 5-4  
    size, 5-4  
command  
    database query, 4-3  
    definition, 4-3  
    query, 4-3  
    using, 4-3  
CommandType property, 4-3  
Common Language Runtime  
    definition, 1-1  
Common Language Runtime Host for Oracle  
    Database, 1-2  
Common Language Runtime Service  
    starting, 7-1  
Configure Your Connection window, 7-7  
connect, 5-1  
connect project, 3-2  
connect\_Click() method, 3-11, 3-13, 4-11, 4-12, 6-8  
Connection, 3-18  
connection  
    add, 5-1, 7-3, 7-8  
    building, 3-13  
    close, 3-18  
    creating, 7-3  
    data, 5-1, 5-10, 7-3  
    data source name, 5-2, 7-4, 7-8  
    detail, 5-2  
    details, 7-4  
    dispose, 4-12  
    hr. (Local Database), 5-10  
    hr.(Local Database), 5-3, 7-5  
    hr.(LocalDatabase), 7-11  
    name, 5-2, 7-4  
    new, 7-3, 7-7

- opening, 3-13
- password, 5-2, 7-4, 7-8
- role, 5-2
- saving password, 5-10
- specific user name and password, 5-2
- SYSDBA, 7-8
- table, 5-10
- test, 5-2, 7-4, 7-8
- test failure, 5-3, 7-5
- test success, 5-2
- user name, 5-2, 7-4
- user name and password, 7-4
- username, 7-8
- connection element, 3-5
- constraint
  - add, 5-7, 5-8
- constraint properties, 5-7
- Constraints tab, 5-7
- copy project, 4-1
- CREATE TABLE, 5-5
- cultural convention, 8-1
- cultural conventions, 8-1
- culture conventions, 8-7
- Culture ID parameter, 8-6
- Culture parameter, 8-6
- CultureInfo, 8-2
- culture-sensitive data, 8-7
- CurrentCulture parameter, 8-12

## D

---

- data
  - navigate, 5-13
  - sort, 5-13
- Data Access for .NET Classes, 1-2
- data connection, 5-1, 5-10
- data entry element, 3-5
- Data Grid, 4-9
- data grid, 4-11, 4-13, 6-9
- data provider, 3-2
  - Oracle Data Provider for .NET, 1-2
- data source name, 5-2, 7-4
- database error message, 3-16
- DataGrid, 4-8
- DataGrid class, 5-11, 6-9
  - filling, 5-12
  - properties, 5-11
- DataReader class, 4-7, 4-8
- DataSet class, 4-8, 4-9, 4-10, 4-11, 4-12
  - definition, 4-8
  - generate, 5-10
  - updating, 8-11
- DataSource, 5-11
- DataSource parameter, 5-11
- date
  - format, 8-2
- date format, 8-2
  - change, 8-11
  - long, 8-2
  - settings, 8-2

- short, 8-2
- standard, 8-2
- Debug menu, 3-14
- default role, 5-2, 7-4
- DELETE statement, 4-7
- deleting data, 4-13
- design property, 4-4
- Design view, 3-5, 4-4, 4-9, 5-3, 5-5, 5-10, 6-6, 6-7
- Design window, 5-10, 5-11
- Designer window, 5-10
- dialog
  - New Project, 7-3
  - new project, 3-1, 3-2
- Direction property, 4-6
- disable button, 3-13, 3-14
- display schema, 5-2, 7-4
- Dispose() method, 3-17, 3-18

## E

---

- Easy Connect
  - syntax, 3-12
- edit package body, 6-5
- error handling, 3-14
  - Try-Catch-Finally, 3-15
- type
  - exceptions with ODP.NET, 3-15
  - ODP.NET, 3-15
  - Oracle, 3-14
- error message, 8-5
- Error property, 3-15
- event
  - click, 3-11
- Events icon, 3-11
- Exception class, 3-17
- ExecuteReader() method, 4-4
- EZ Connect Adapter, 3-12

## F

---

- FCL, 1-1
- File menu, 3-1, 3-5, 7-3
- finally block, 4-12
- foreign key, 5-7
- Framework Class Libraries
  - definition, 1-1

## G

---

- GetClientInfo() method, 8-7
- GETCURSORS method, 6-3
- GETCURSORS procedure, 6-5
- GETDEPARTMENTNO procedure, 7-11
- getDepartmentno(), 7-5, 7-6
- getDepartmentno() method, 7-5, 7-10
- GetSessionInfo() method, 8-8
- GetThreadInfo() method, 8-12
- global application
  - deployment, 8-1
  - development, 8-1
  - .NET framework, 8-1



- planning, 8-1
- global applications
  - introduction, 8-1
- globalization
  - definition, 8-1
- globalization session information, 8-10
- globalization support, 8-7
  - client, 8-7
  - ODP for .NET, 8-7
- globalization support features, 8-1

## H

---

- handling errors, 3-14
- host, 3-12
- HR, 4-1, 4-2
- HR\_Connect\_CS folder, 4-1, 4-2
- HR\_Connect\_CS project, 3-2, 4-9
- HR\_Connect\_CS solution, 4-2
- HR\_Connect\_VB folder, 4-1
- HR\_Connect\_VB project, 3-2, 4-9
- HR\_Connect\_VB solution, 4-2
- HR\_DATA package, 6-4, 6-5, 6-6
- HR\_DataSet\_ODP\_CS project, 4-9, 6-2
- HR\_DataSet\_ODP\_VB project, 4-9
- HR\_DeployStored\_CS project, 7-3, 7-10
- HR\_DeployStored\_VB project, 7-3, 7-10
- HR\_Globalization\_CS project, 8-8
- HR\_Globalization\_VB project, 8-8
- HR\_ODP\_CS folder, 4-1, 4-2
- HR\_ODP\_CS solution, 4-2
- HR\_ODP\_VB folder, 4-1
- HR\_ODP\_VB solution, 4-2
- HR\_ODT\_CS project, 5-1
- HR\_ODT\_VB project, 5-1
- HR\_StoredProcedure\_CS project, 6-2
- HR\_StoredProcedure\_VB project, 6-2

## I

---

- Imports statement, 3-4
- index
  - add, 5-5
  - add key, 5-6
  - creating, 5-5
  - key, 5-6
  - name, 5-5
  - property, 5-5
- Indexes tab, 5-5
- initial programmatic statement, 3-4
  - C#, 3-5
  - VB, 3-5
- InitializeComponent() method, 5-12
- INSERT statement, 4-7
- inserting data, 4-13
- InstallShield wizard, 2-2

## L

---

- Label, 3-6
- label, 4-4

- language
  - sorting rules, 8-2
- lightning icon, 3-11
- linguistic sort, 8-4
- list box, 4-5
- ListBox, 4-4, 4-8
- literal quotation marks, 4-5
- local user convention, 8-2
- locale
  - definition, 8-1
  - synchronizing, 8-6
- locale awareness, 8-1
- locale characteristics
  - definition, 8-1
- localization
  - definition, 8-2
  - resources, 8-2
- LTER TABLE, 5-8

## M

---

- memory address, 6-2
- memory location, 6-2
- menu
  - Build, 7-6
  - build, 7-7
  - Debug, 3-14
  - File, 3-5, 7-3
  - file, 3-1
  - Project, 3-3
  - View, 3-2, 3-4, 3-5, 3-6, 3-8, 3-10, 4-4, 4-9, 5-1
- method
  - add, 6-2
  - Add(), 4-6
  - connect\_Click(), 3-13, 6-8
  - detail, 7-10
  - Dispose(), 3-17, 3-18
  - GETCURSORS, 6-3
  - getDepartmentno(), 7-10
  - name, 6-3
  - Open(), 3-13
  - parameter, 6-3
  - save\_Click(), 5-13
  - type, 6-3
- method parameter
  - add, 6-3
  - binding, 6-8
  - data type, 6-3
  - definition, 6-8
  - detail, 6-3
  - direction, 6-3
  - name, 6-3
- Microsoft Development Environment window, 5-10
- Microsoft internationalization
  - URL, 8-1
- Microsoft .NET Framework
  - definition, 1-1

## N

---

- Name property, 3-9, 3-11, 4-9

- name property, 4-4
- navigate
  - data, 5-13
  - records, 5-13
- .NET assembly, 1-2
- .NET framework, 8-2
- .NET languages, 1-2
- .NET stored functions and procedures
  - creating, 7-5
  - deploying, 7-7
  - running, 7-11
- .NET Stored Procedures, 2-1
- .NET stored procedures, 1-2
  - deployment, 7-1
- .NET Types, 8-12
- New Package window, 6-2, 6-4, 6-5
- new project, 3-1
- New Project dialog, 3-1, 3-2, 7-3
- NLS error messages setting, 8-6
- NLS number format
  - settings, 8-3
- NLS Registry, 8-8
- NLS sort order, 8-4
- NLS\_LANG, 8-7
- NLS\_LANG parameter, 8-7
- NLS\_LANGUAGE parameter, 8-2, 8-3, 8-5, 8-6
  - common, 8-6
- NLS\_SORT parameter, 8-4, 8-5
- NLS\_TERRITORY parameter, 8-2, 8-3, 8-4, 8-6
  - common, 8-6
- number format, 8-3

## O

---

- ODBC, 2-1
- ODP.NET
  - definition, 1-2
  - globalization, 8-1
- ODP.NET Types, 8-12
- OLE DB, 2-1
- onnect, 5-1
- OPD.NET
  - using, 5-1
- Open() method, 3-13, 8-7
- opening connection, 3-13
- Oracle, 2-1, 5-1
- Oracle Application Express, 2-1, 7-8
- Oracle Data Provider for .NET, 2-1, 3-4
  - definition, 1-2
  - using, 4-1
- Oracle Database Extensions
  - features
    - Common Language Runtime Host for Oracle Database, 1-2
    - Data Access for .NET Classes through Oracle Data Provider for .NET, 1-2
    - Oracle Deployment Wizard for .NET, 1-2
    - Oracle Deployment Wizard for Visual Studio .NET, 1-2
- Oracle Database XE, 2-1
  - client, 2-1
  - server, 2-1
- Oracle date format, 8-2
- Oracle Deployment Wizard, 7-7, 7-8
- Oracle Deployment Wizard for Visual Studio
  - .NET, 1-2
- Oracle Developer Tools
  - definition, 1-2
  - features
    - designer, 1-2
    - drag and drop, 1-2
    - dynamic help, 1-2
    - Oracle Data Window, 1-2
    - Oracle Explorer, 1-2
    - Oracle Query Window, 1-2
    - PL/SQL editor, 1-2
    - Solution Explorer, 3-3
    - wizard, 1-2
  - installation, 2-2
  - testing installation, 2-4
- Oracle Developer Tools for Visual Studio .NET
  - using, 5-1
- Oracle error message, 8-5
- Oracle Explorer, 5-1, 5-3, 5-8, 5-10, 6-2, 6-5, 6-6, 7-3, 7-5, 7-11
- Oracle Globalization Registry, 8-8
- Oracle globalization setting, 8-7
- Oracle globalization settings, 8-8
- Oracle linguistic sort, 8-4
- Oracle number format, 8-3
- Oracle project
  - creating, 7-2
- OracleCommand, 4-6
  - using stored procedure, 6-8
- OracleCommand class, 4-3, 4-4, 4-6
- OracleConnection, 3-11
- OracleConnection class, 3-12, 5-10, 8-7, 8-8
  - GetSessionInfo() method, 8-8
  - Open() method, 8-7
- Oracle.DataAccess, 3-4
- Oracle.DataAccess.dll, 3-3
- OracleDataAccess.dll, 3-2
- OracleDataAdapter class, 5-10
- OracleDataReader class, 4-4, 4-8, 4-9
- OracleDbType property, 4-6
- OracleError class, 3-14, 3-15
- OracleErrorCollection class, 3-15
- OracleException class, 3-15, 3-17
- OracleGlobalization, 8-7
- OracleGlobalization class, 8-8, 8-12
  - GetClientInfo() method, 8-7
  - GetThreadInfo() method, 8-12
  - SetThreadInfo() method, 8-12
- OracleGlobalization.GetClientInfo(), 8-7
- OracleGlobalization.GetClientInfo() method, 8-7
- OracleParameter class, 4-6, 6-8, 6-9
- OracleParameterCollection class, 4-6
- OracleRefCursor class, 6-2
- OracleServiceXE, 3-15, 3-17
- OracleXEClrAgent, 7-2

OracleXEClrAgent service, 7-1  
order  
    ascending, 5-13  
    descending, 5-13  
Output window, 6-6, 7-7

## P

---

package  
    edit body, 6-5  
    HR\_DATA, 6-2, 6-4, 6-5, 6-6  
    name, 6-2  
    new, 6-2  
    save, 6-6  
package body, 6-1  
PACKAGE BODY type, 6-1  
package interface, 6-1  
package specification, 6-1  
PACKAGE type, 6-1  
parameter, 6-6  
parameter detail, 6-7  
ParameterName property, 4-6  
password, 3-12  
    save, 5-2, 5-10  
    saving, 7-4  
    set, 7-8  
PasswordChar property, 3-10  
PL/SQL package  
    body, 6-1  
    definition, 6-1  
    interface, 6-1  
    specification, 6-1  
PL/SQL package bodies, 6-1  
PL/SQL packages  
    introduction, 6-1  
PL/SQL stored procedure  
    definition, 6-1  
    in ODP.NET, 6-8  
    introduction, 6-1  
    ref cursor, 6-2  
PL/SQL stored procedures, 6-1  
port, 3-12  
preview SQL, 5-5, 5-6, 6-4  
Preview SQL window, 5-5, 5-6, 5-8, 6-4  
primary key, 5-7  
    column, 5-8  
procedure  
    GETCURSORS, 6-5  
    GETDEPARTMENTNO, 7-11  
    run, 7-11  
    value, 6-6, 6-7  
project  
    add reference, 3-2  
    connect, 3-2  
    connection, 3-2  
    copy, 4-1  
    HR\_Connect\_CS, 3-2, 4-9  
    HR\_Connect\_VB, 3-2, 4-9  
    HR\_DataSet\_ODP\_CS, 4-9, 6-2  
    HR\_DataSet\_ODP\_VB, 4-9

HR\_DeployStored\_CS, 7-3, 7-10  
HR\_DeployStored\_VB, 7-3, 7-10  
HR\_ODT\_CS, 5-1  
HR\_ODT\_VB, 5-1  
HR\_StoredProcedure\_CS, 6-2  
HR\_StoredProcedure\_VB, 6-2  
location, 3-2, 7-3  
name, 3-2, 7-3  
new, 3-1, 7-2  
solution, 3-2  
template, 3-2, 7-3  
type, 3-2, 7-3  
    Visual Basic, 3-2, 7-3  
    Visual C#, 3-2, 7-3  
Project menu, 3-3  
propert  
    text, 4-9  
properties  
    data, 5-11  
Properties window, 3-8, 3-10, 3-11, 4-4, 4-5, 4-9, 4-10, 5-11  
property, 3-7, 3-9, 3-10, 4-9  
    appearance, 3-9, 4-9  
    behavior, 3-10  
    BindByName, 4-6  
    design, 3-9, 3-11, 4-4, 4-9, 4-10  
    Direction, 4-6  
    Error, 3-15  
    Name, 3-9, 3-11, 4-9  
    name, 4-4  
    OracleDbType, 4-6  
    OracleDbType, 4-6  
    ParameterName, 4-6  
    PasswordChar, 3-10  
    Size, 4-6  
    Text, 3-7, 3-9, 3-10, 4-4  
    Value, 4-6

## Q

---

query performance, 4-5  
query work area  
    definition, 6-2

## R

---

record, 4-13, 5-9  
    add, 5-9  
    delete, 4-14  
    navigate, 5-13  
    sort, 5-13  
REF cursor, 6-1  
ref cursor  
    accessibility, 6-2  
    assigning, 6-5  
    database round trip, 6-2  
    database update, 6-2  
    definition, 6-2  
    introduction, 6-1  
    PL/SQL datatype, 6-2  
    PL/SQL stored procedure, 6-2

- reading, 6-2
- read-only, 6-2
- serial, 6-2
- reference
  - add, 3-3
  - adding, 3-2
  - folder, 3-4
  - list, 3-3
- rename solution, 4-2
- resources
  - deploying, 8-2
  - localization, 8-2
  - packaging, 8-2
- result set, 6-2
- Retrieving, 4-5
- retrieving data
  - accessor type methods, 4-5
  - bind variable, 4-5, 4-6, 4-7
  - looping, 4-8
  - multiple column, 4-8
  - multiple row, 4-8
  - multiple values, 4-7
  - native data type, 4-5
  - native Oracle data type, 4-5
  - simple query, 4-4
  - value, 4-5
  - zero-based ordinal, 4-5
- role, 7-4
  - user, 5-2
  - user default, 5-2
- row, 5-9
- row set, 6-2
- run
  - application, 4-8, 5-14
- Run application, 3-17
- run application, 3-14, 3-16, 4-5, 4-12
- Run Function window, 7-11
- Run Procedure window, 6-6
- running an application, 5-12

## S

---

- Sample Data, 2-1
- sample data, 2-1
- Sample Schema, 2-1
- Save button, 5-13
- Save icon, 3-5
- save\_Click() method, 4-10, 4-12, 5-13, 5-14
- schema
  - content, 5-3
  - display, 5-2, 7-4
- schema object, 1-2, 7-4
- SELECT statement, 4-5, 4-6
  - bind variable, 4-6
  - simple, 4-6
- service\_name, 3-12
- services, 7-1
- session globalization parameter, 8-8
- session globalization setting, 8-8
- SetThreadInfo() method, 8-12

- Setup file, 2-2
- Setup icon, 2-2
- simple query, 4-4
- Size, 4-6
- Size property, 4-6
- software download, 2-2
- solution, 3-2
  - deploy, 7-7
- Solution 'HR\_Connect\_CS', 4-2
- Solution 'HR\_Connect\_VB', 4-2
- Solution Explorer, 3-2, 3-3, 3-4, 3-5, 4-2
- sort
  - data, 5-13
  - records, 5-13
- Specify an assembly and library name window, 7-9
- Specify copy options window, 7-9
- Specify methods and security details window, 7-10
- Specify your deployment option window, 7-8
- SQL
  - preview, 5-5
- SQL injection attack, 4-5
- SQL preview, 5-6
- SQL query, 4-3
- SQL statement, 4-5, 4-6, 5-6
- SQL statement string, 4-3
- SQLPlus, 8-2
- Start Database window, 3-17
- statement
  - Case, 3-17
  - DELETE, 4-7
  - Imports, 3-4
  - INSERT, 4-7
  - memory, 4-5
  - optimizing, 4-5
  - parsing, 4-5
  - reusing, 4-5
  - UPDATE, 4-7
  - using, 3-4
- stored procedure, 4-6
  - definition, 6-1
  - run, 6-6
- stored procedure declaration, 4-6
- Summary window
  - window
    - Summary, 7-11
- System.Globalization, 8-2
- System.Resources, 8-2
- System.Threading.Thread.CurrentThread.CurrentCulture parameter, 8-12

## T

---

- table
  - add constraint, 5-7
  - add data, 5-8
  - constraint, 5-6, 5-8
    - add, 5-7, 5-8
  - association, 5-7
  - local column, 5-7
  - primary key, 5-7

- referenced column, 5-7
- constraint name, 5-7
- constraint properties, 5-7
- constraint type, 5-7
- creating, 5-3
- data, 5-8
- grid, 5-9
- name, 5-3
- new, 5-3
- new relational, 5-3
- query, 5-9
- record, 5-9
- relational, 5-3
- retrieve data, 5-8
- row, 5-9
- simple query, 5-9
- SQL form, 5-5
- table design view, 5-6, 5-8
- table design window, 5-3
- table grid, 5-9
- Test, 7-8
- test
  - result, 7-8
- test result window, 7-8
- Text Box, 3-8
- Text property, 3-7, 3-9, 3-10, 4-4, 4-9
- thread-based globalization setting, 8-12
- Toolbox, 3-6, 3-7, 3-8, 3-10, 4-4, 4-5, 4-9, 5-11
- try code block, 4-3, 4-11, 6-8
- Try-Catch-Finally, 3-15, 3-17
- Try-Catch-Finally error handling, 3-15
- type of project, 3-2

## U

---

- unlocking account, 5-2, 7-4
- unlocking user account, 2-4
  - Oracle XE Database interface, 2-4, 7-8
- update event, 5-13
- UPDATE statement, 4-7
- updating data, 4-13
  - bind variable, 4-7
- user
  - default role, 7-4
  - id, 3-12
  - locale settings, 8-6
  - role, 5-2, 7-4
- user's locale, 8-1
- user\_source view, 6-1
- using statement, 3-4

## V

---

- Value property, 4-6
- variable declaration, 4-10
- VB statement
  - Imports, 3-4
- View, 4-9
- view
  - Code, 3-4
  - code, 5-12, 5-13, 6-8

- Design, 3-5, 4-4, 4-9, 5-3, 5-5, 5-10, 6-6, 6-7
- table design, 5-6, 5-8
  - user\_source, 6-1
- View menu, 3-2, 3-4, 3-5, 3-6, 3-8, 3-10, 4-4, 4-9, 5-1
- Visual Basic, 3-2, 7-3
- Visual C#, 3-2, 7-3
- Visual Studio .NET 2003, 2-2

## W

---

- warning, 3-15
- WHERE statement clause, 4-5
- window
  - Add Connection, 5-1, 7-4, 7-8
  - Add Method, 6-3
  - Design, 5-10, 5-11
  - Designer, 5-10
  - Microsoft Development Environment, 5-10
  - New Package, 6-2, 6-4, 6-5
  - Output, 6-6, 7-7
  - Preview SQL, 5-5, 5-6, 5-8, 6-4
  - Properties, 3-8, 3-10, 3-11, 4-4, 4-5, 4-10, 5-11
  - Run Function, 7-11
  - Run Procedure, 6-6
  - Specify an assembly and library name, 7-9
  - Specify copy options, 7-9
  - Specify methods and security details, 7-10
  - Specify your deployment option, 7-8
  - table design, 5-3
  - test result, 7-8
- Windows Explorer, 4-1
- Windows forms, 3-6, 3-8, 3-10, 4-4, 4-8, 4-9, 5-11
- Windows Registry, 8-7

## X

---

- XE Common Language Runtime agent, 7-1

## Z

---

- zero-based ordinal, 4-5

