

Software Engineering Lab

Building .NET Applications on the Oracle Database with Microsoft Visual Studio

Lab 1 – Connected Layer

Lab Rules

- ❑ You MUST attend in your section.
 - ❑ Please commit to the lab start time.
 - ❑ No attendance exceptions from TAs.
 - ❑ Attendance exceptions only signed from doctors.
-

Agenda

- ❑ Required installations
- ❑ Introduction to ODP.Net
- ❑ ODP.Net Object Model
- ❑ DVD Rental Case Study
- ❑ Running DB Script
- ❑ Data Retrieval using OracleDataReader
 - Actors Form Design
 - Populating ComboBox with ActorIDs in the Form_Load

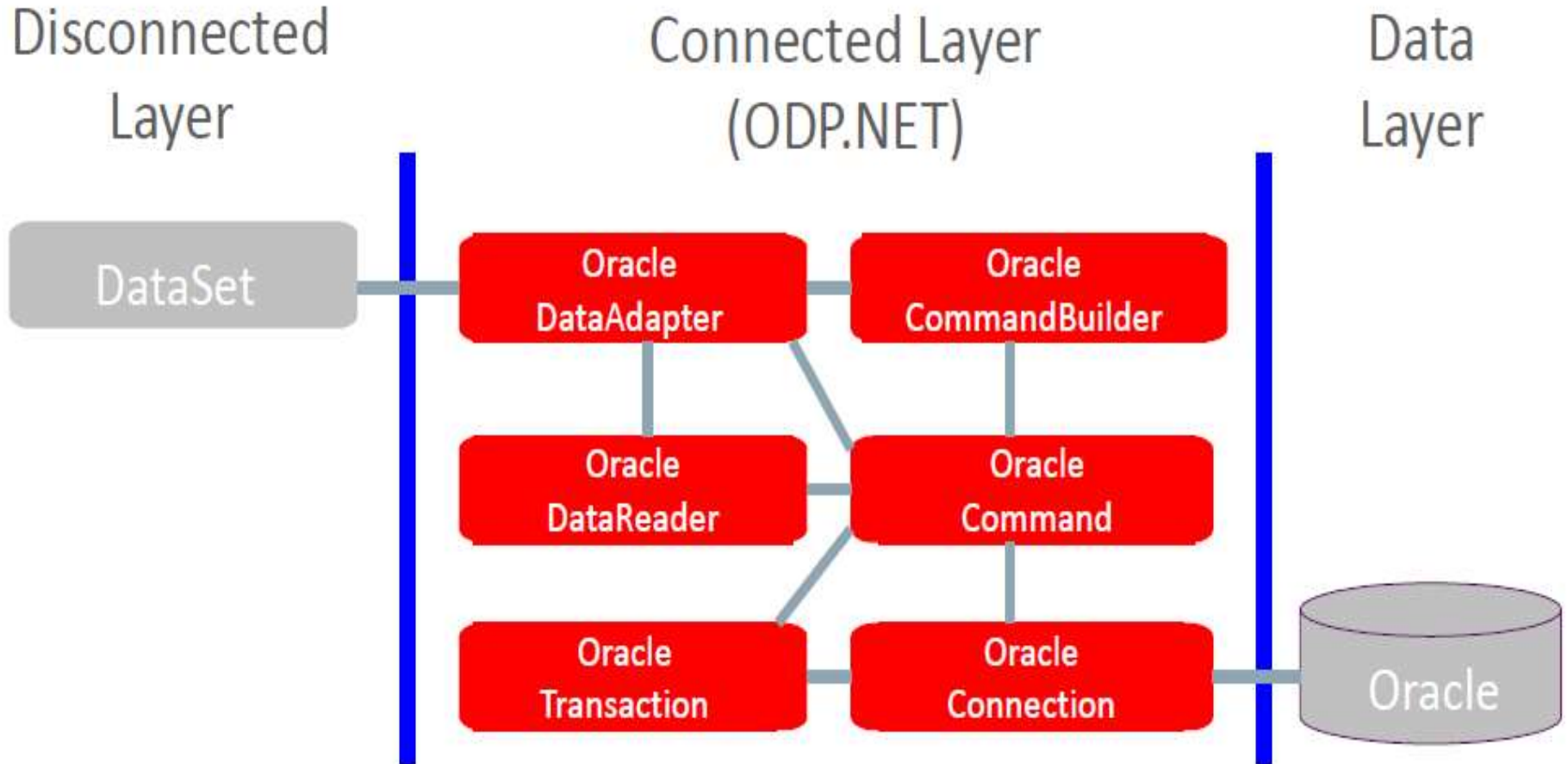
Required installations

- Following Products are needed in order to build a VS.Net application on Oracle DB:
 - Oracle Database 11g
 - Microsoft Visual Studio.Net (Visual C#)

Introduction to ODP.Net

- Oracle Data Provider for .NET (ODP.NET) provides fast and efficient ADO.NET data access from .NET client applications to Oracle databases.
- Use **ODP** (from Oracle) instead of ADO (from Microsoft).
- Note that the *installation of Oracle Database includes Oracle Data Provider for .NET*

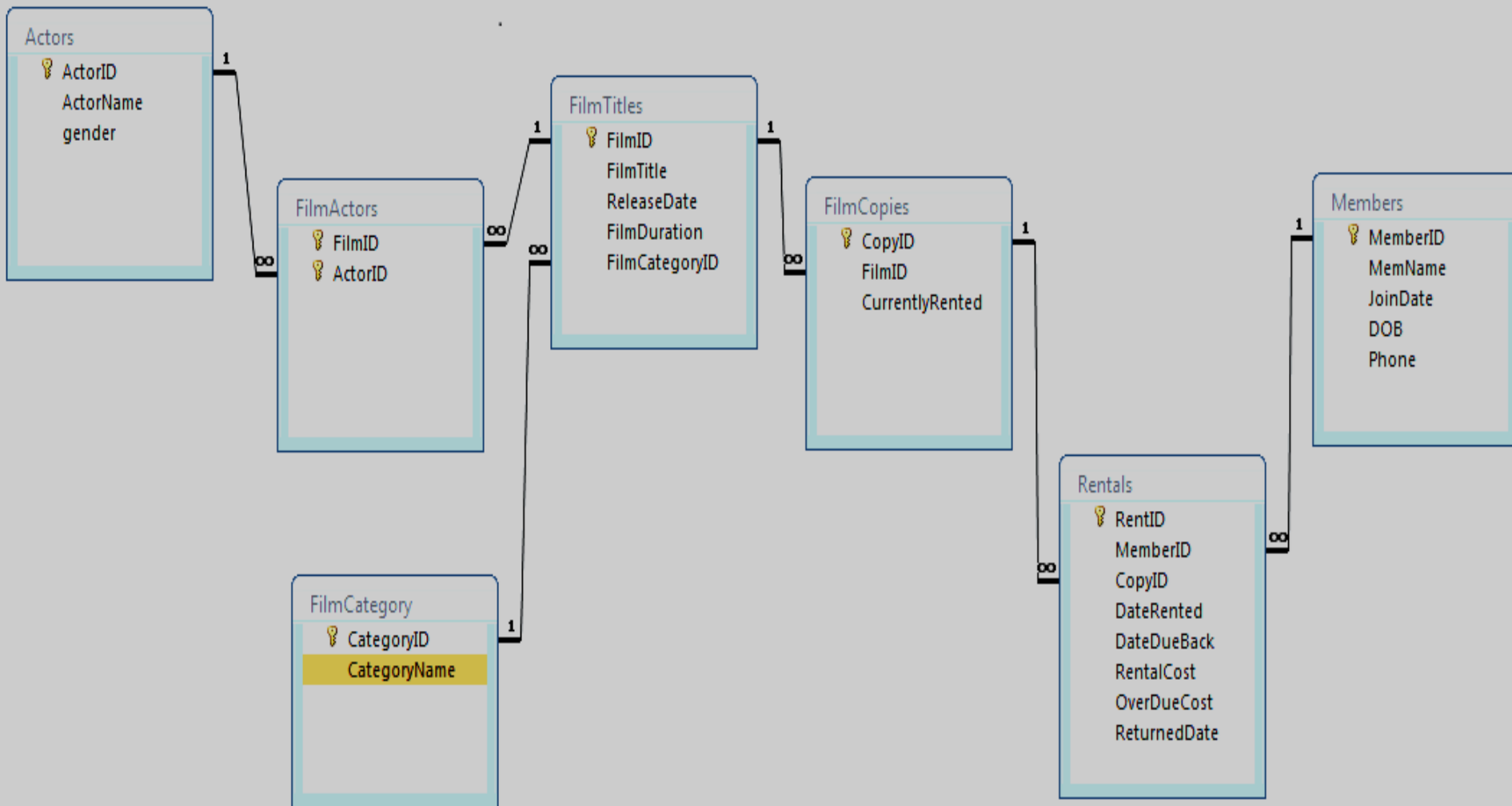
ODP.Net Object Model



DVD Rental Case Study

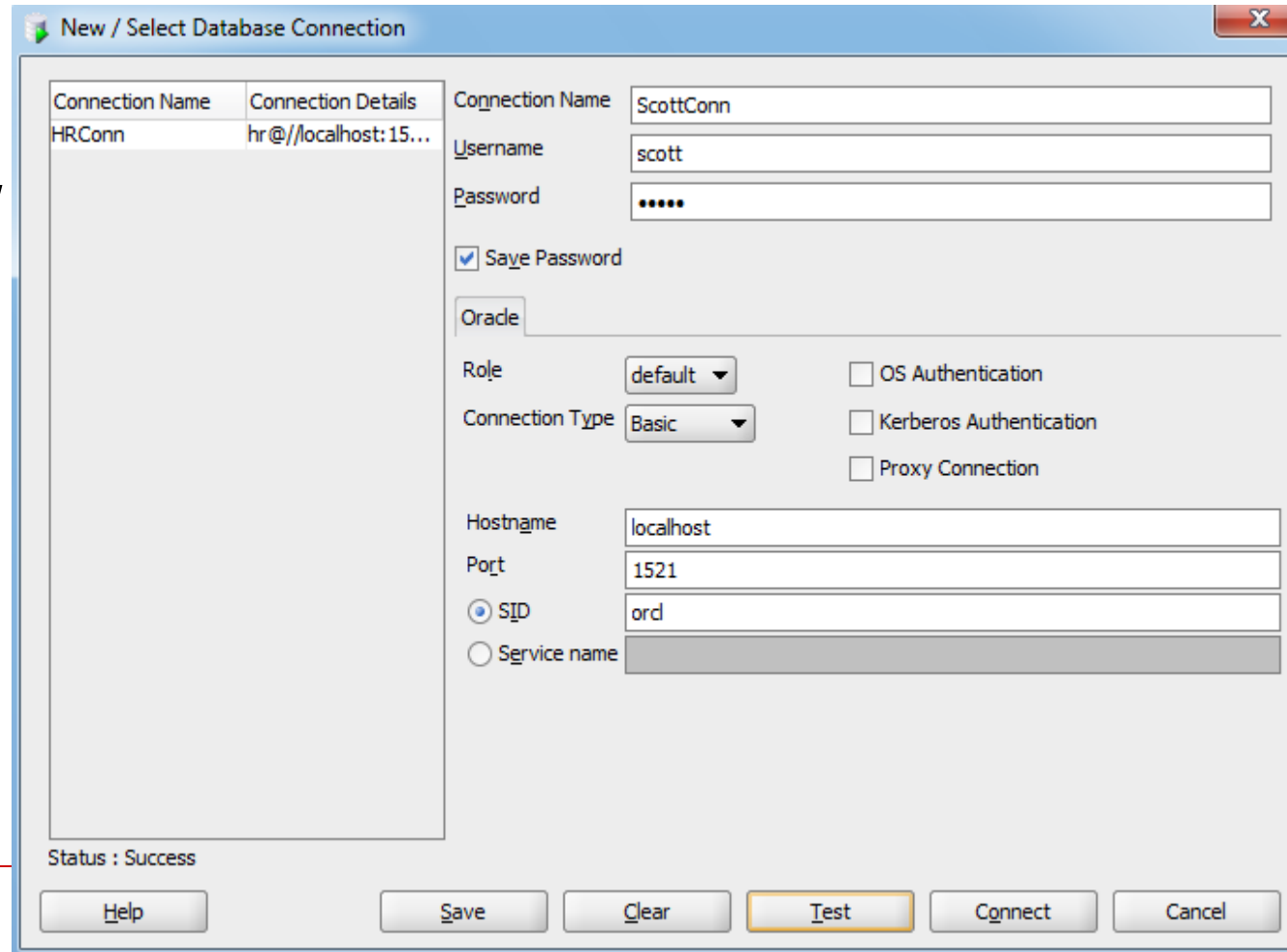
- The following data model is designed to hold information related to a DVD rentals store.
 - The Tables required should include:
 - Members
 - Rentals
 - FilmCopies
 - FilmTitles
 - FilmCategory
 - FilmActors
-

DB Design



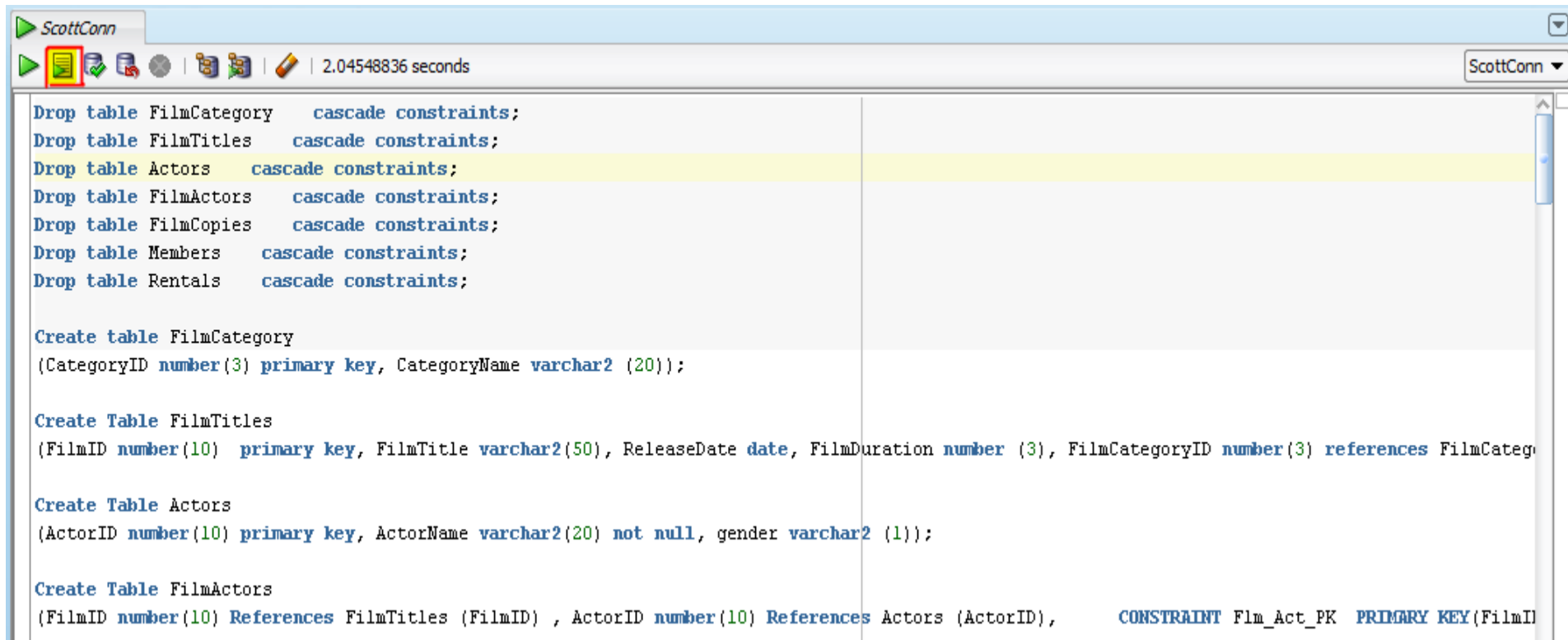
Running DB Script

1. Open SQL Developer
2. Create a new connection for **scott** user with the password **tiger**



Running DB Script

3. Copy the DB script from “**DVD rental Script**” file, and paste it into the SQL Developer
4. Press “**Run Script**” button (or press **F5**)



The screenshot shows the SQL Developer interface with a script titled "ScottConn". The script contains SQL commands to drop and create tables for a DVD rental database. The "Run Script" button (a green play icon) is highlighted in the toolbar. The script text is as follows:

```
Drop table FilmCategory cascade constraints;
Drop table FilmTitles cascade constraints;
Drop table Actors cascade constraints;
Drop table FilmActors cascade constraints;
Drop table FilmCopies cascade constraints;
Drop table Members cascade constraints;
Drop table Rentals cascade constraints;

Create table FilmCategory
(CategoryID number(3) primary key, CategoryName varchar2 (20));

Create Table FilmTitles
(FilmID number(10) primary key, FilmTitle varchar2(50), ReleaseDate date, FilmDuration number (3), FilmCategoryID number(3) references FilmCategory);

Create Table Actors
(ActorID number(10) primary key, ActorName varchar2(20) not null, gender varchar2 (1));

Create Table FilmActors
(FilmID number(10) References FilmTitles (FilmID) , ActorID number(10) References Actors (ActorID),
```

CONSTRAINT Flm_Act_PK PRIMARY KEY(FilmID, ActorID))

Pre-requisites

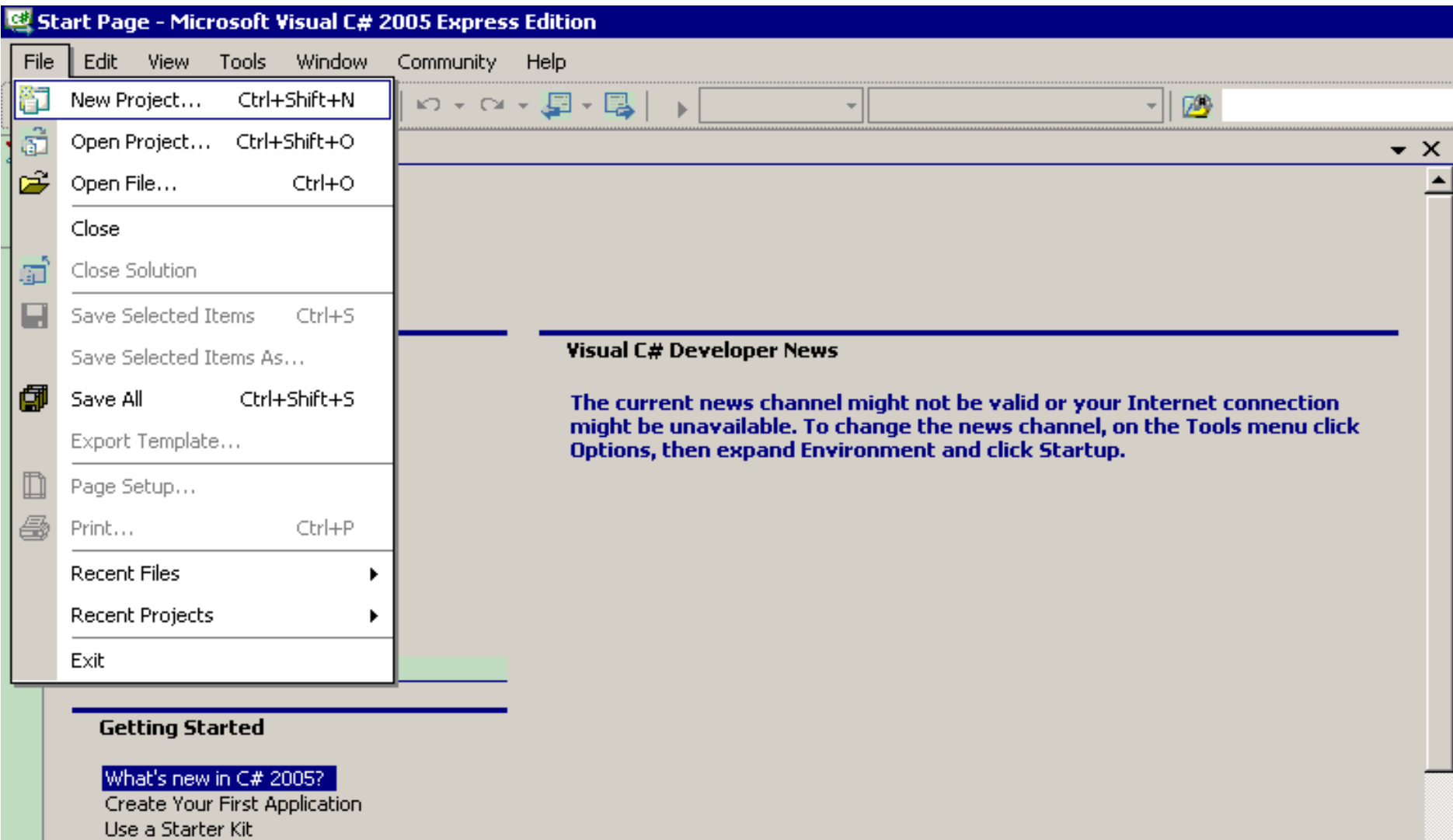
1. Install Oracle 11g Database
2. Install Visual Studio.Net
3. Run the DVD Rental database creation script
4. You should be aware of all SQL statements (select, insert, update, delete)

Connected Layer Actors form example

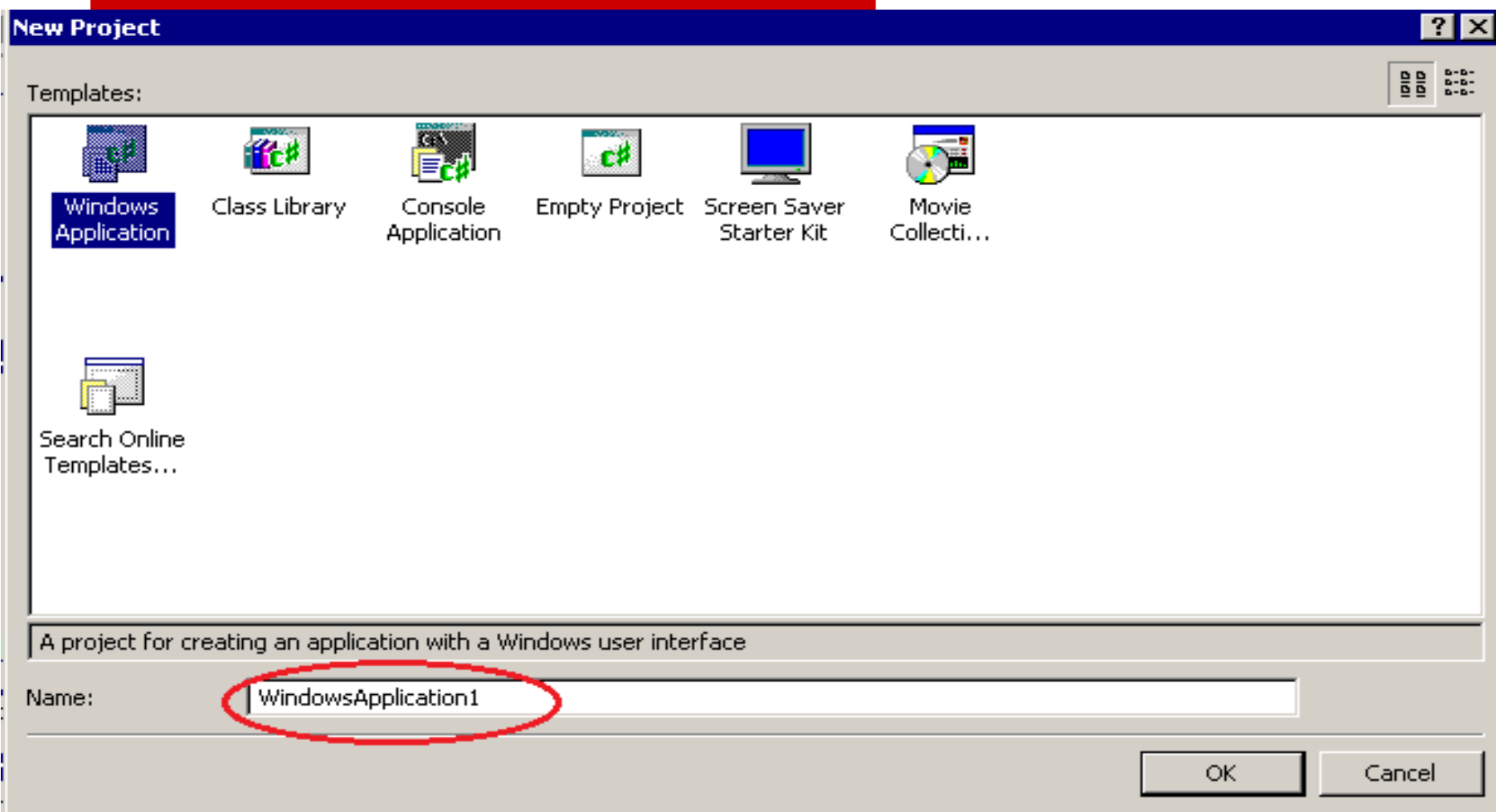
#Connecting to Oracle DB from C

1. Creating VS.Net C# project
2. Adding a Reference to Oracle library
3. Adding controls to the Form Design
4. Writing the code

1- Creating VS.Net C# Project



Creating VS.Net C# Project

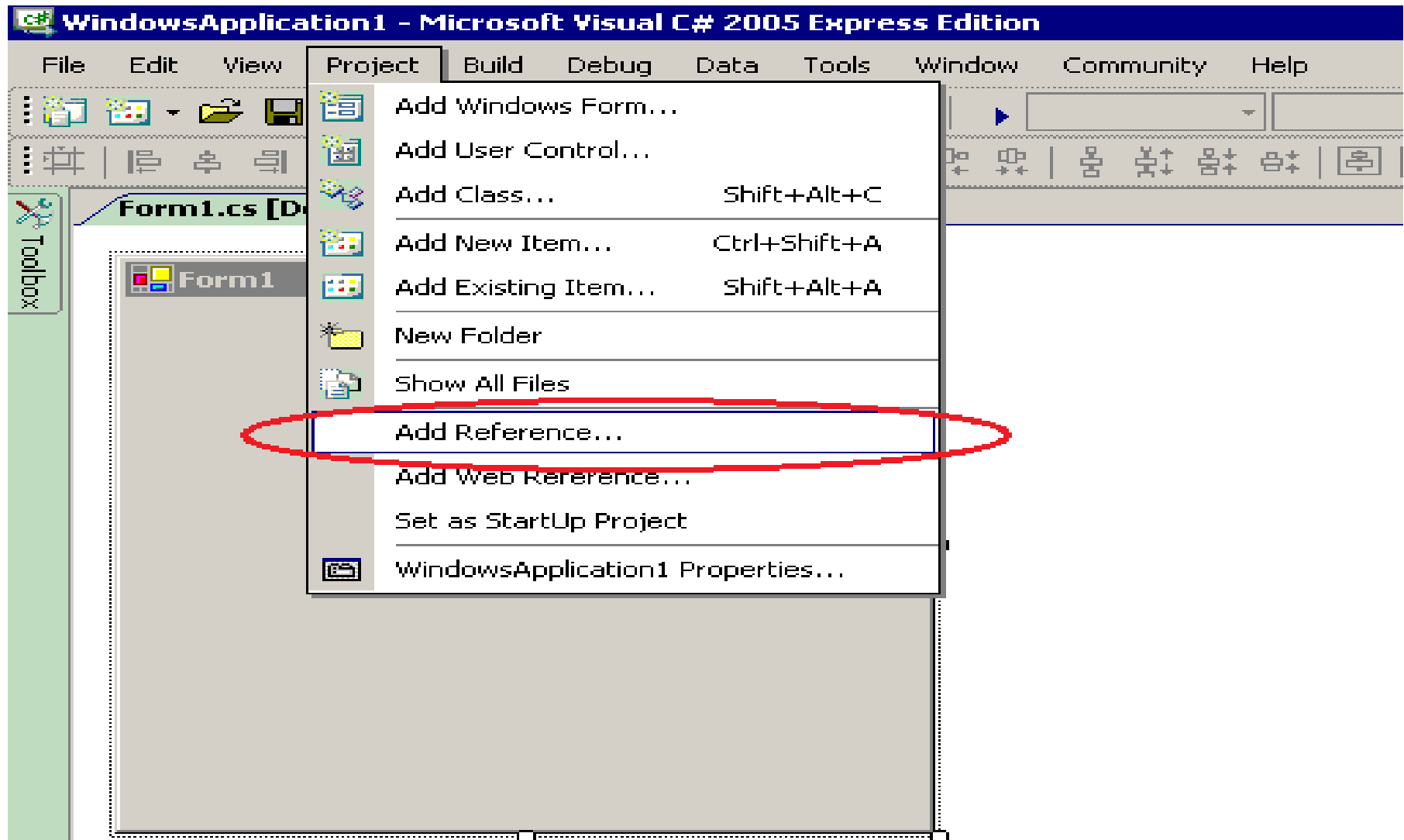


2- Adding a Reference

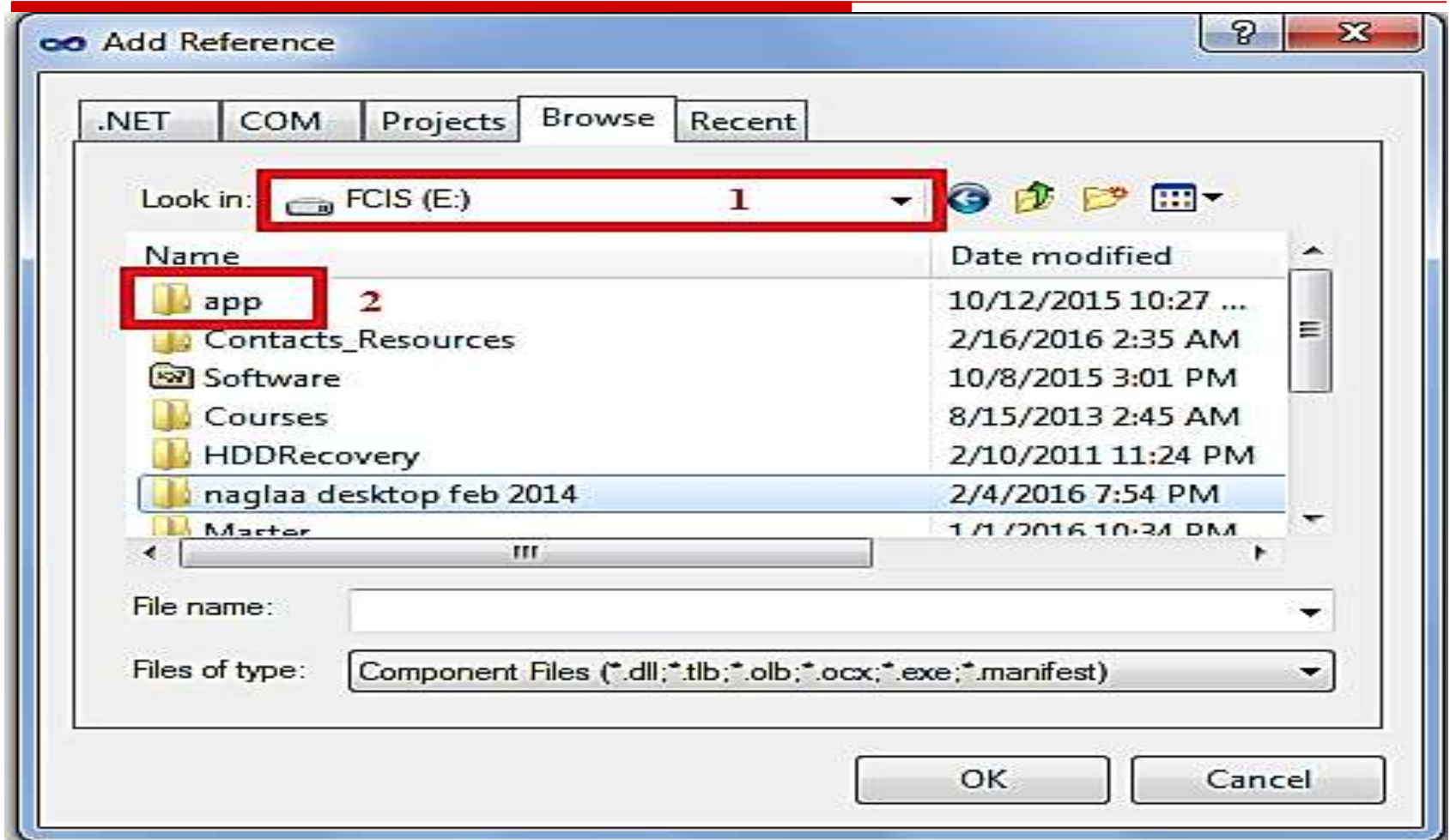
- To connect the project to an Oracle database, you must add a reference to the **Oracle.DataAccess.dll**, which contains the data provider.
- This library is located at:

E:\app\pc_user\product\11.2.0\dbhome_1
\ODP.NET\bin\2.x

2- Adding a Reference (Cont.)

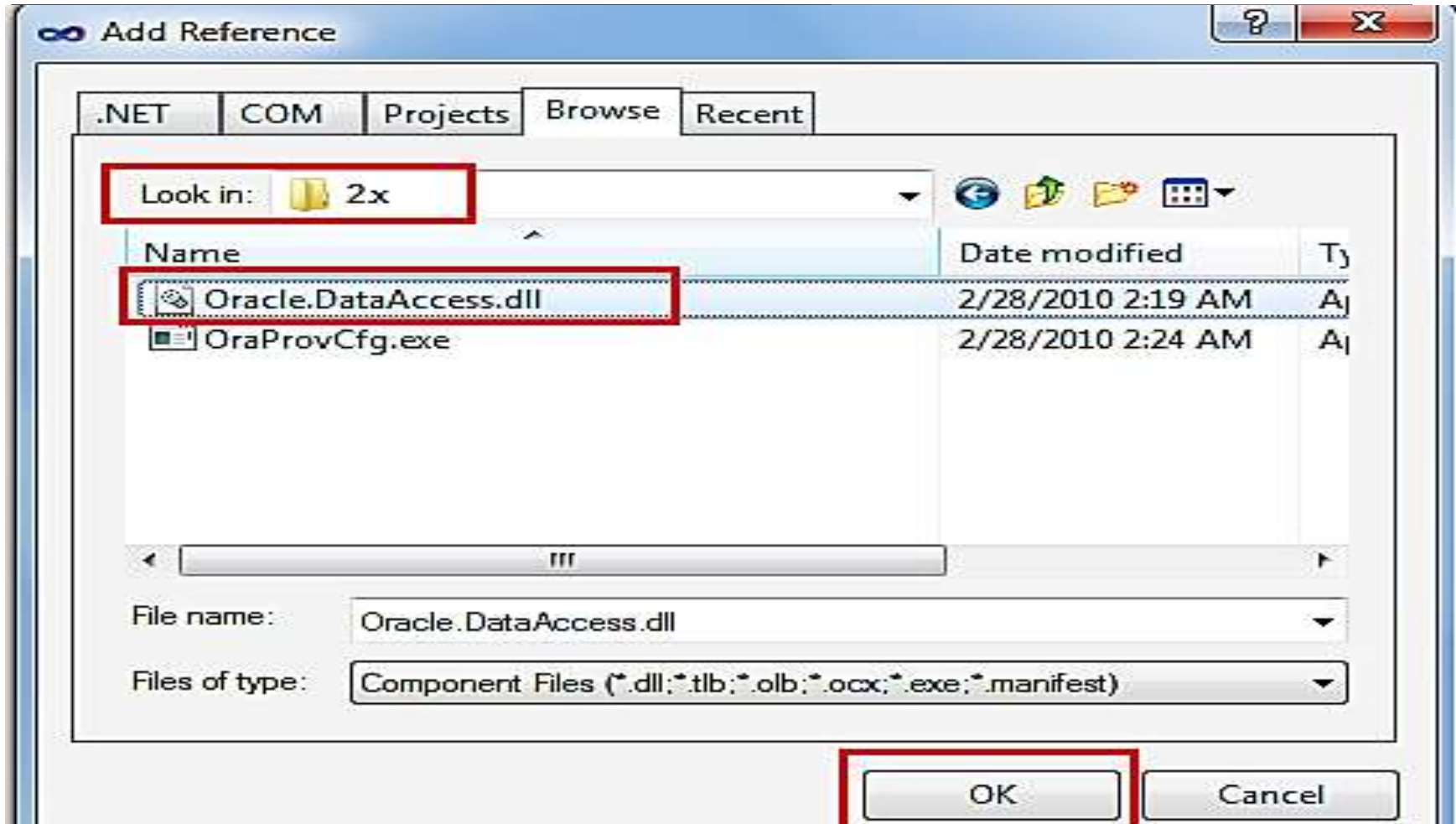


2- Adding a Reference (Cont.)

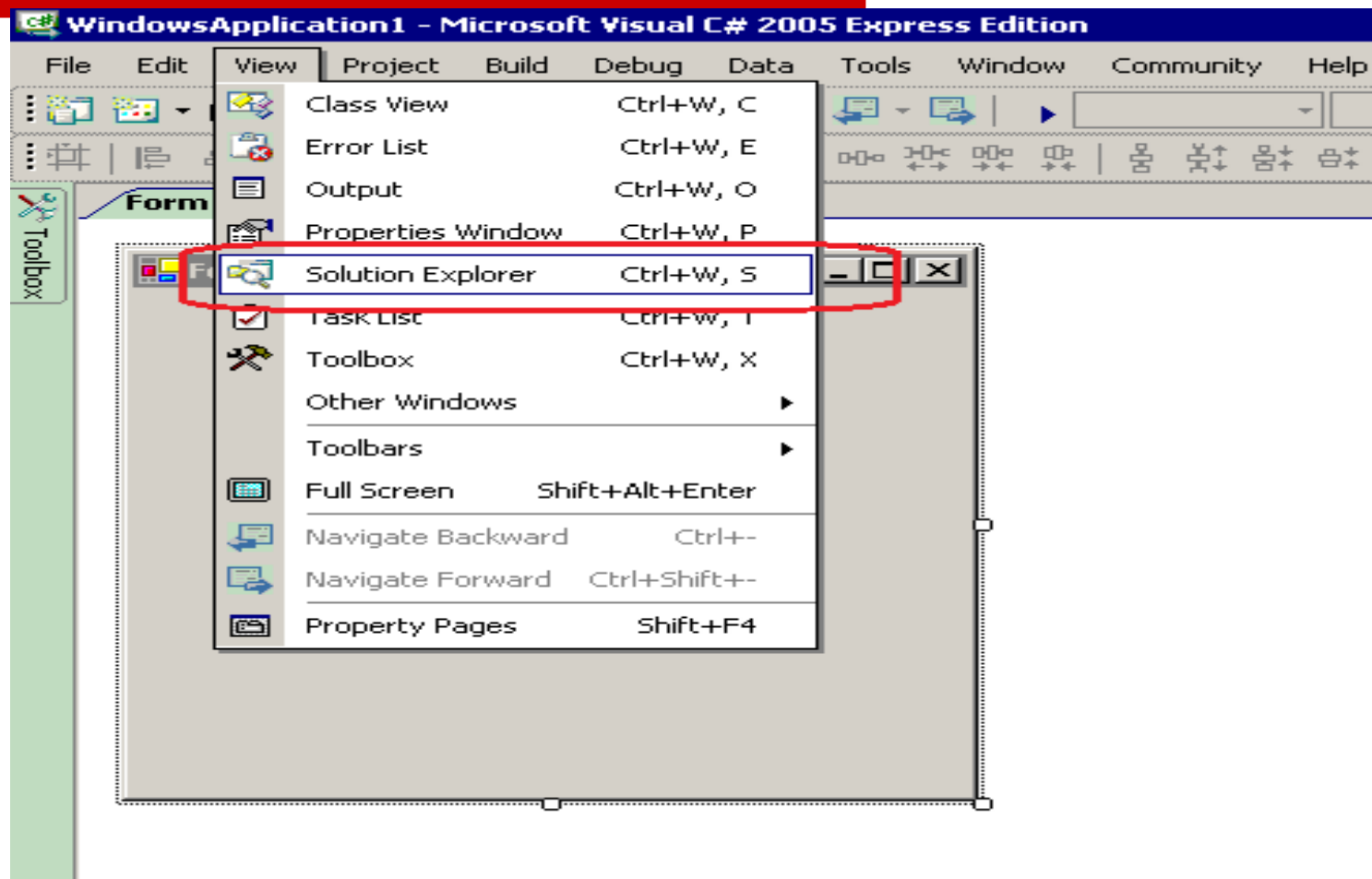


E:\app\pc_user\product\11.2.0\dbhome_1\ODP.NET\bin\2.x

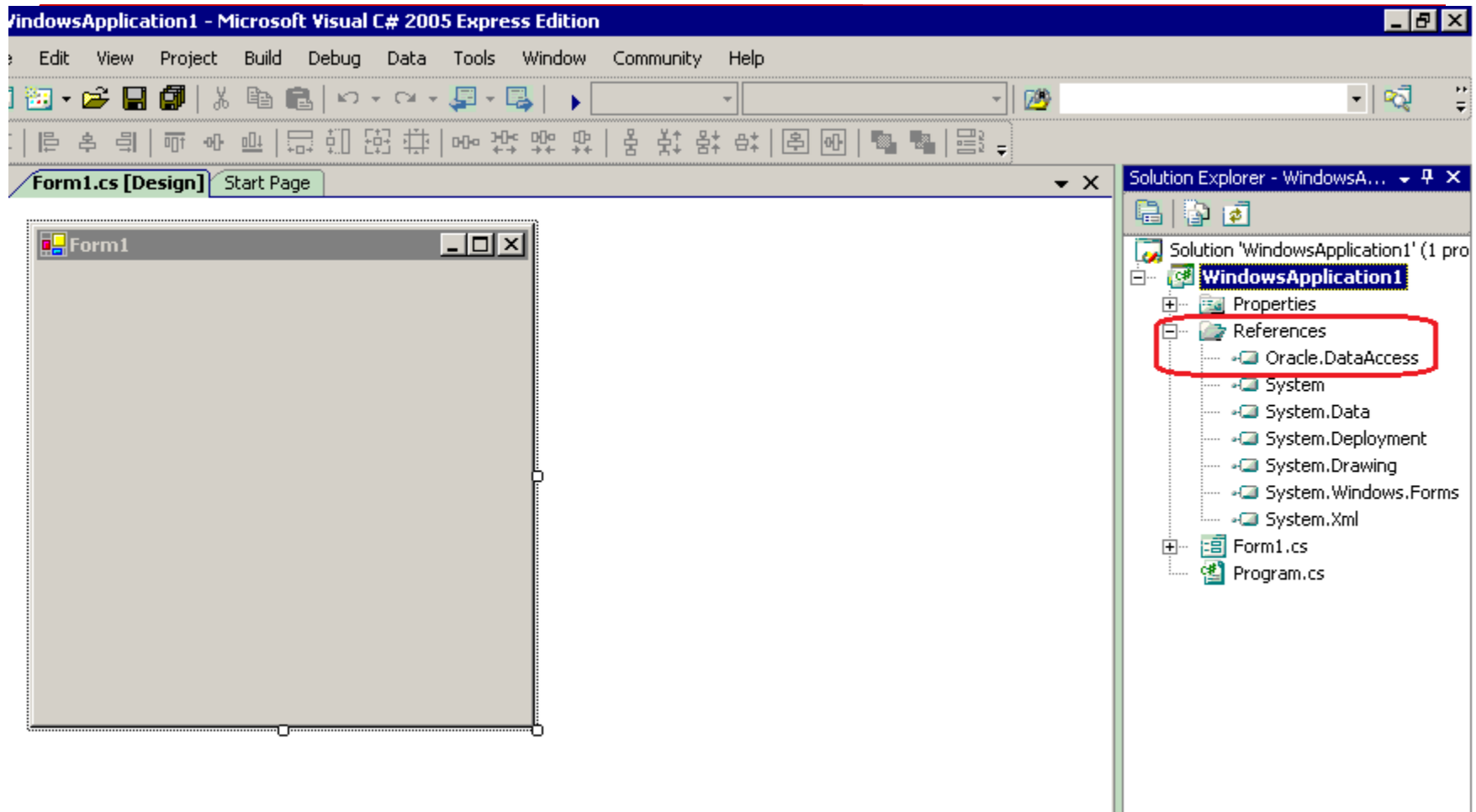
2- Adding a Reference (Cont.)



2- Adding a Reference (Cont.)



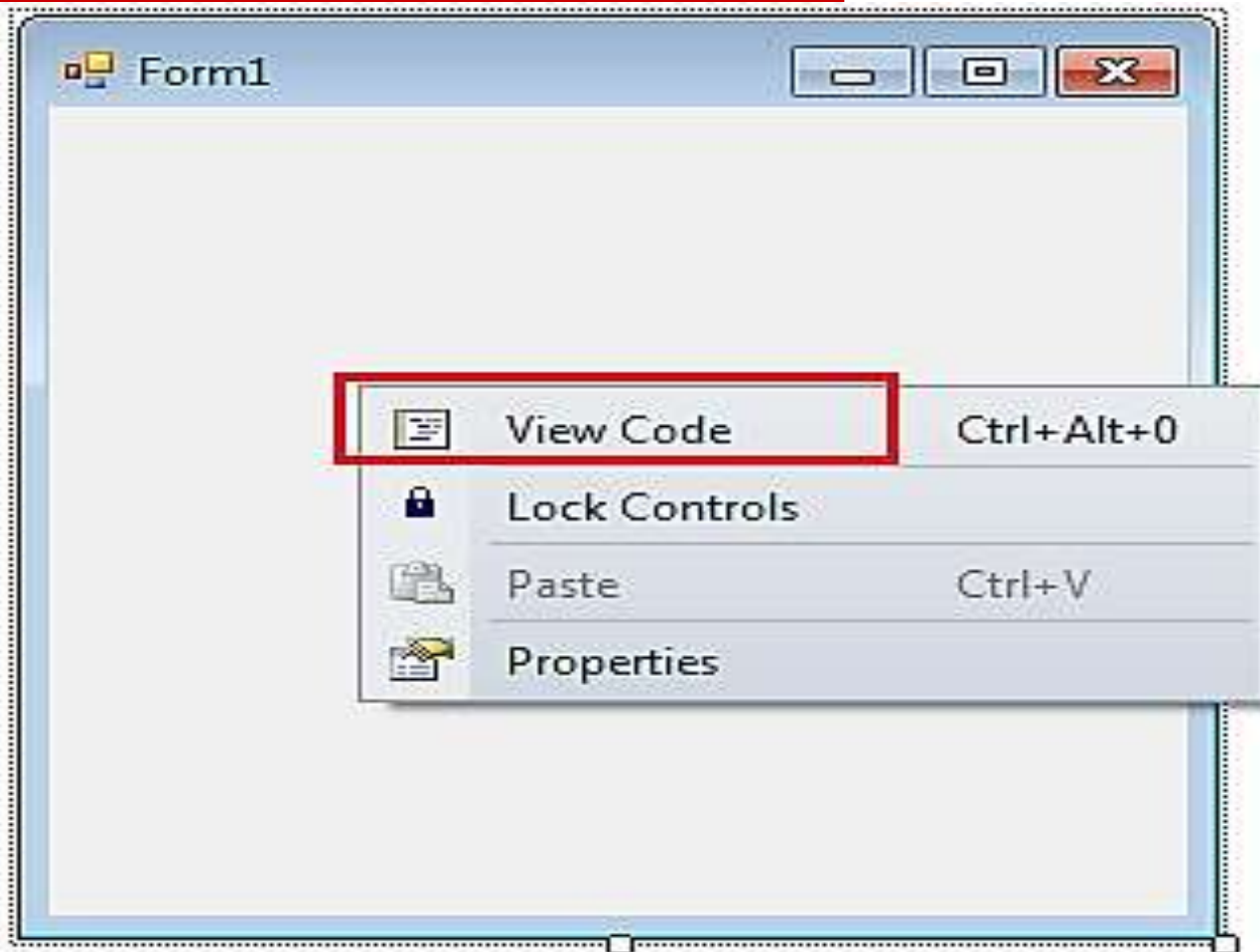
2- Adding a Reference (Cont.)



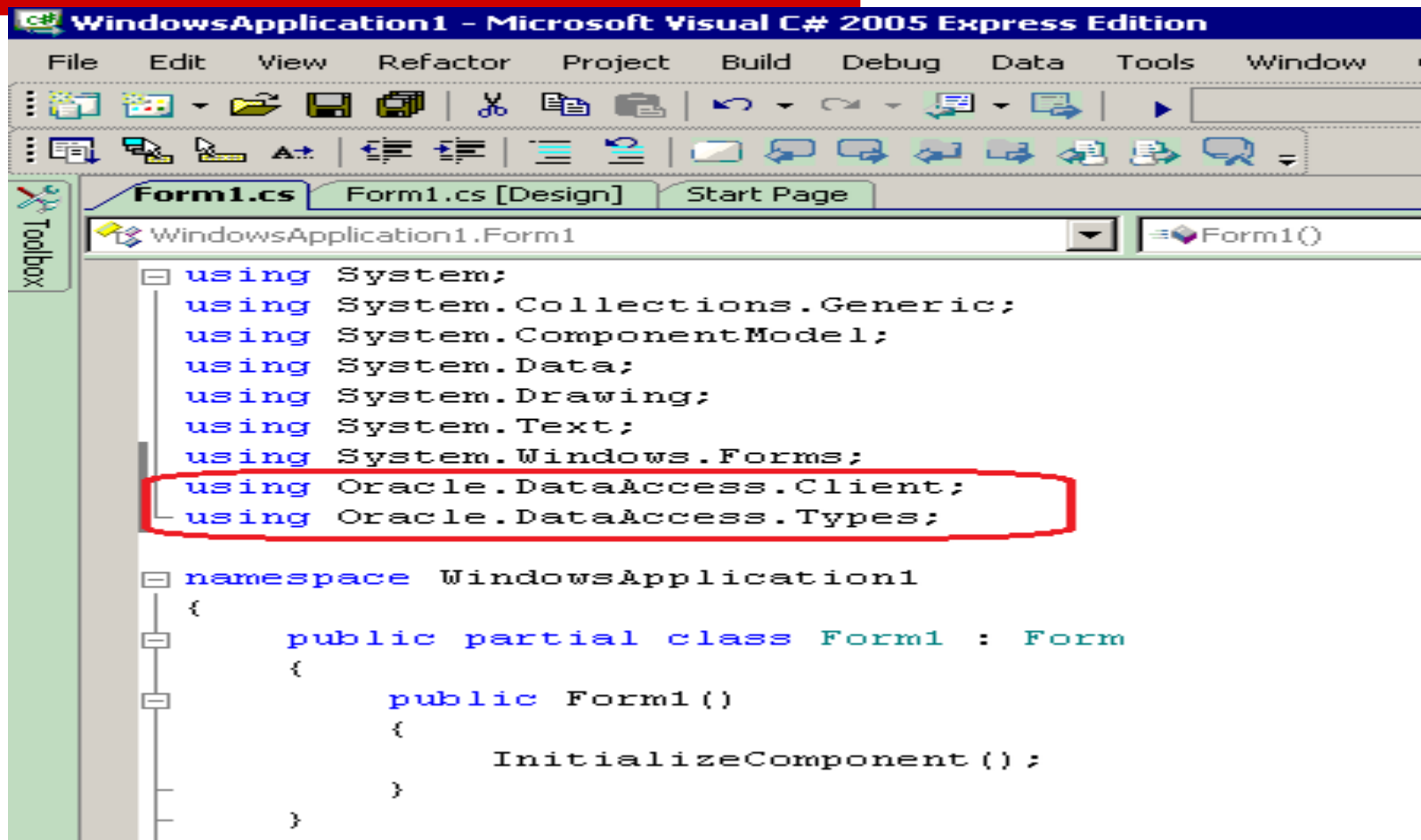
Oracle.DataAccess.dll Library

- The Oracle.DataAccess.dll provides two namespaces:
 - The *Oracle.DataAccess.Client* namespace contains ODP.NET classes and enumerations for the client-side provider.
 - The *Oracle.DataAccess.Types* namespace contains the Oracle Data Provider for .NET data types (ODP.NET Types).

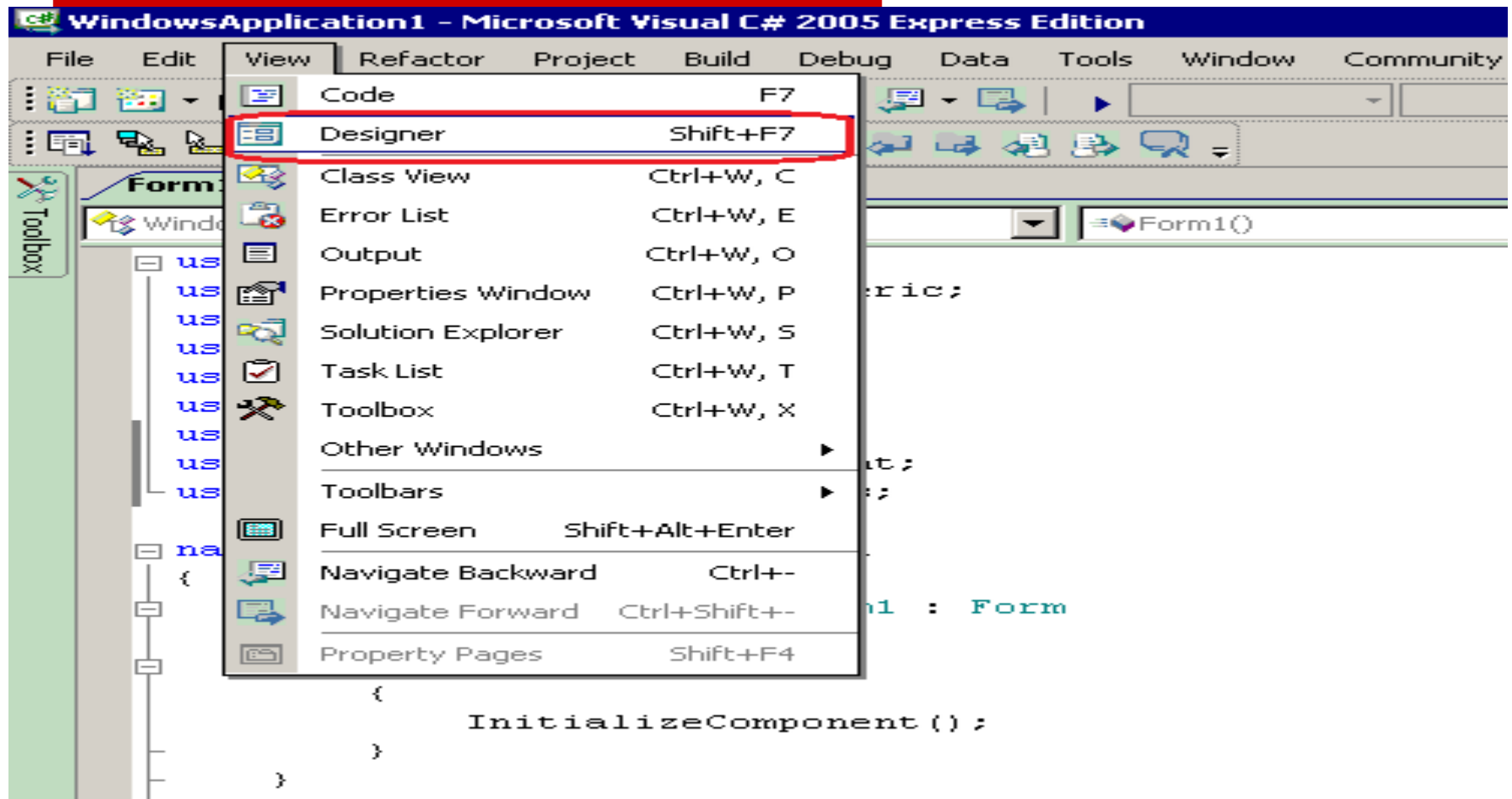
Adding Initial C# Statements



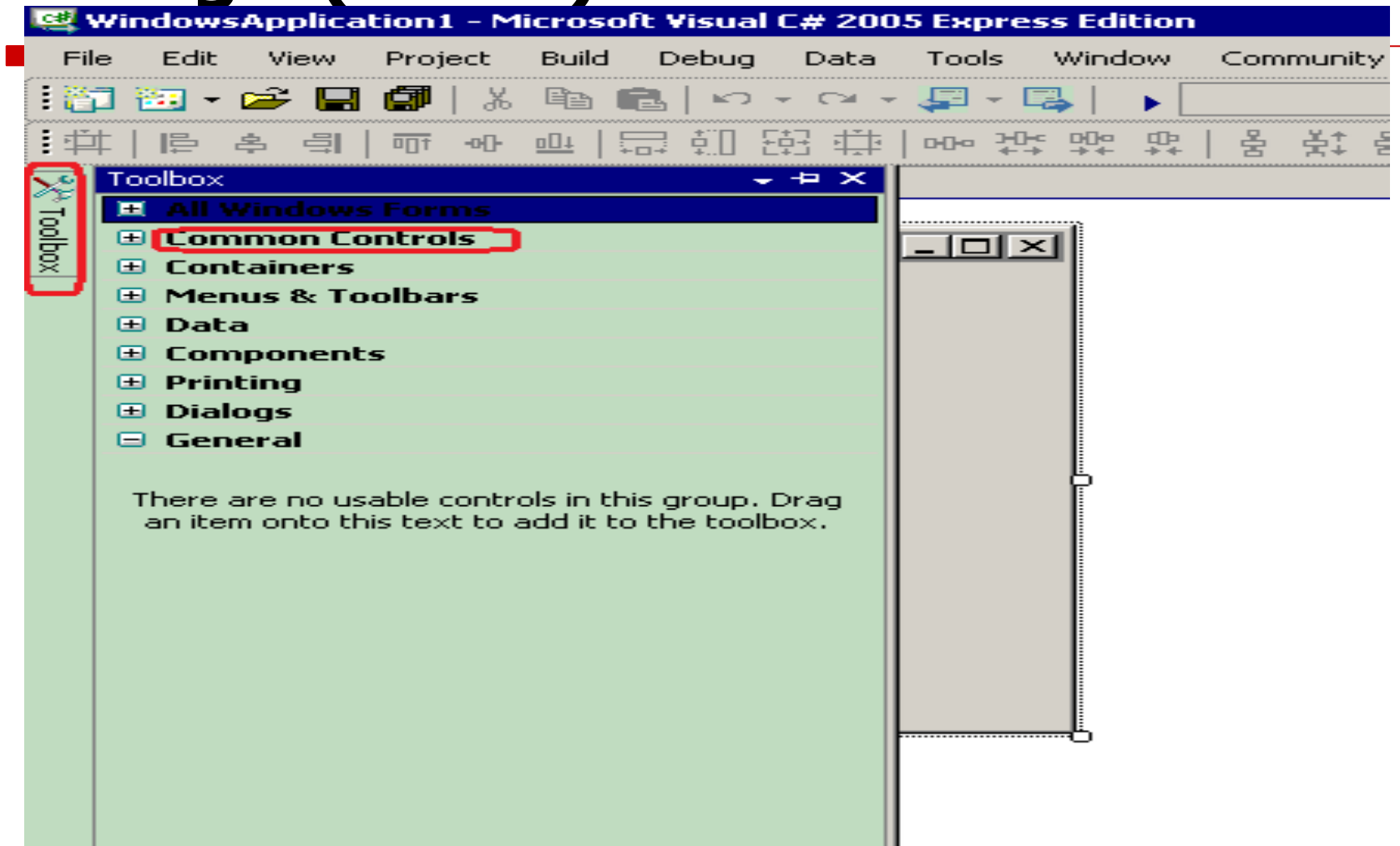
Adding Initial C# Statements



3- Adding controls to the Form Design



3- Adding controls to the Form Design (Cont.)



3- Adding controls to the Form Design (Cont.)



4. Writing the code

1. Creating OracleConnection object
 2. Creating OracleCommand object
 - Connection
 - Command text
 - Command Type
 - Parameters
 - Execute command
 3. Closing the connection (after executing all commands)
-

1- Form Preparation

a. Fill Actors IDs

(Select Statement)

Creating Oracle Connection Object

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

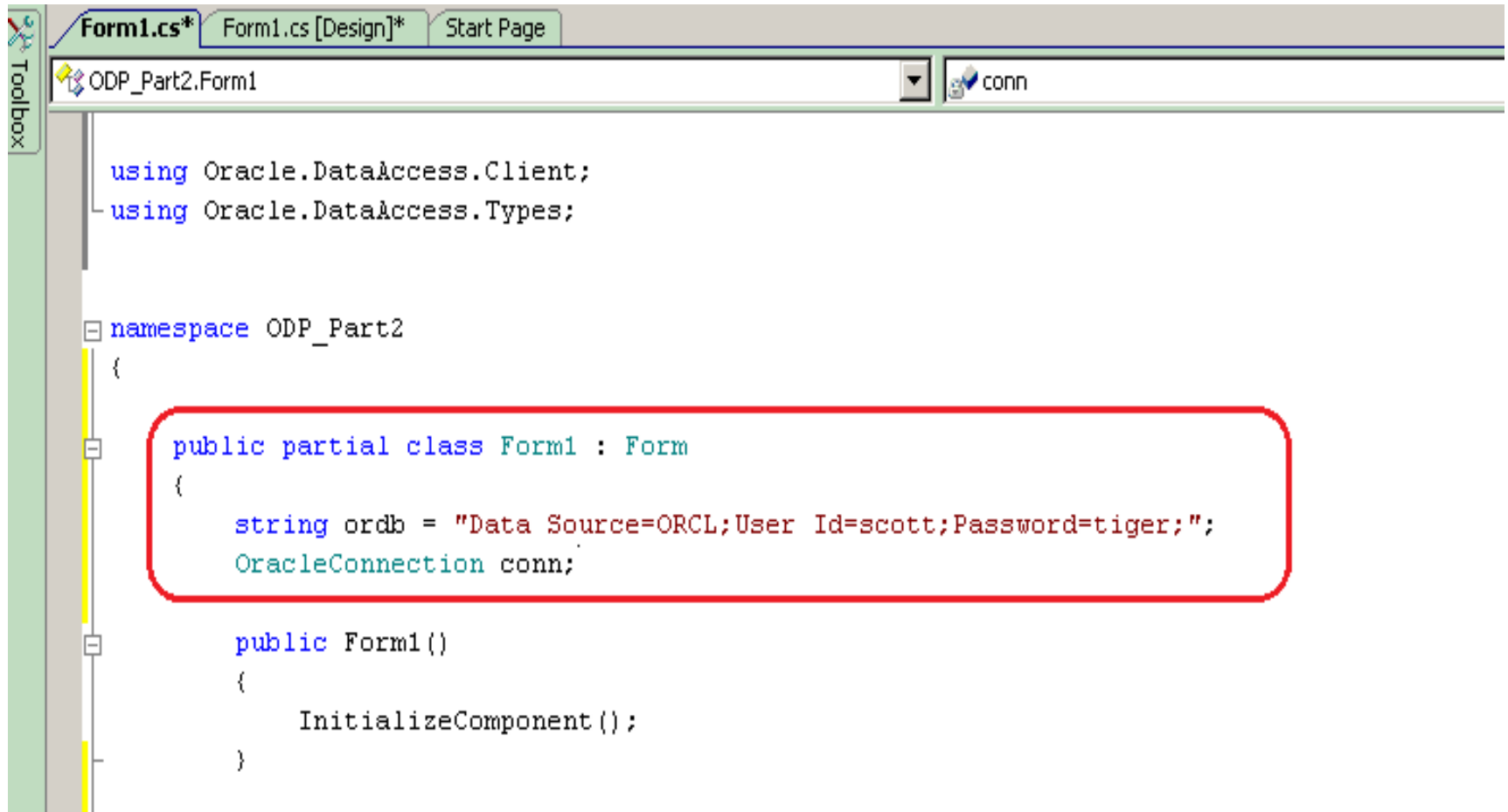
- Define the global variables

```
string ordb = "Data source=orcl;User Id=scott;  
    Password=tiger;";  
OracleConnection conn;
```

- Then open the connection

```
conn = new OracleConnection(ordb) ;  
conn.Open();
```

Declaring *Global* Connection Object



Creating Connection Object(Cont.)

Form_Load event

```
public partial class ActorsForm : Form
{
    string ordb = "Data source=orcl;User Id=scott; Password=tiger;";
    OracleConnection conn;

    public ActorsForm()
    {
        InitializeComponent();
    }

    private void ActorsForm_Load(object sender, EventArgs e)
    {
        conn = new OracleConnection(ordb);
        conn.Open();
    }
}
```


Creating OracleCommand object

- The command object is used to specify the SQL command text that is executed, either a SQL string or a stored procedure.
- It creates a database request, sends the request to the database, and returns the result.

Creating OracleCommand object (Cont.)

- To create OracleCommand Object, write this code in the **Form load** event handler:

```
OracleCommand cmd = new OracleCommand();  
cmd.Connection = conn;  
cmd.CommandText = "select ActorID from Actors";  
cmd.CommandType = CommandType.Text;
```

Creating OracleCommand object (Cont.)

```
private void ActorsForm_Load(object sender, EventArgs e)
{
    conn = new OracleConnection(ordb);
    conn.Open();

    OracleCommand cmd = new OracleCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select ActorID from Actors";
    cmd.CommandType = CommandType.Text;
```

Using OracleDataReader to retrieve data

- An **OracleDataReader** object represents a forward-only, read-only, in-memory result set.
- Unlike the DataSet, the OracleDataReader object stays connected and fetches one row at a time.

Using OracleDataReader to retrieve data (Cont.)

- An OracleDataReader object is constructed by a call to the **ExecuteReader** method of the OracleCommand object.
- To retrieve data from DB, write this code:

```
OracleDataReader dr = cmd.ExecuteReader();  
while (dr.Read())  
{  
    cmb_ID.Items.Add(dr[0]);  
}  
dr.Close();
```

Using OracleDataReader to retrieve data (Cont.)

```
private void ActorsForm_Load(object sender, EventArgs e)
{
    conn = new OracleConnection(ordb);
    conn.Open();

    OracleCommand cmd = new OracleCommand();
    cmd.Connection = conn;
    cmd.CommandText = "select ActorID from Actors";
    cmd.CommandType = CommandType.Text;

    OracleDataReader dr = cmd.ExecuteReader();
    while (dr.Read())
    {
        cmb_ID.Items.Add(dr[0]);
    }
    dr.Close();
}
```

Using OracleDataReader to retrieve data (Cont.)

□ ExecuteReader:

- This method executes a command specified in the CommandText and returns an OracleDataReader object.

□ Read():

- This method reads the next row in the result set.
- To return multiple rows, use loop
- The initial position of the data reader is before the first row. Therefore, the Read method must be called to fetch the first row.
- The row that was just read is considered the *current row*. If the OracleDataReader has no more rows to read, it returns false.

Using OracleDataReader to retrieve data (Cont.)

□ Columns in result set

- Access columns in result set using their indexes (dr[**ColumnIndex**]) or names (dr[**"ColumnName"**])
- **ColumnIndex** is Zero-based column index.

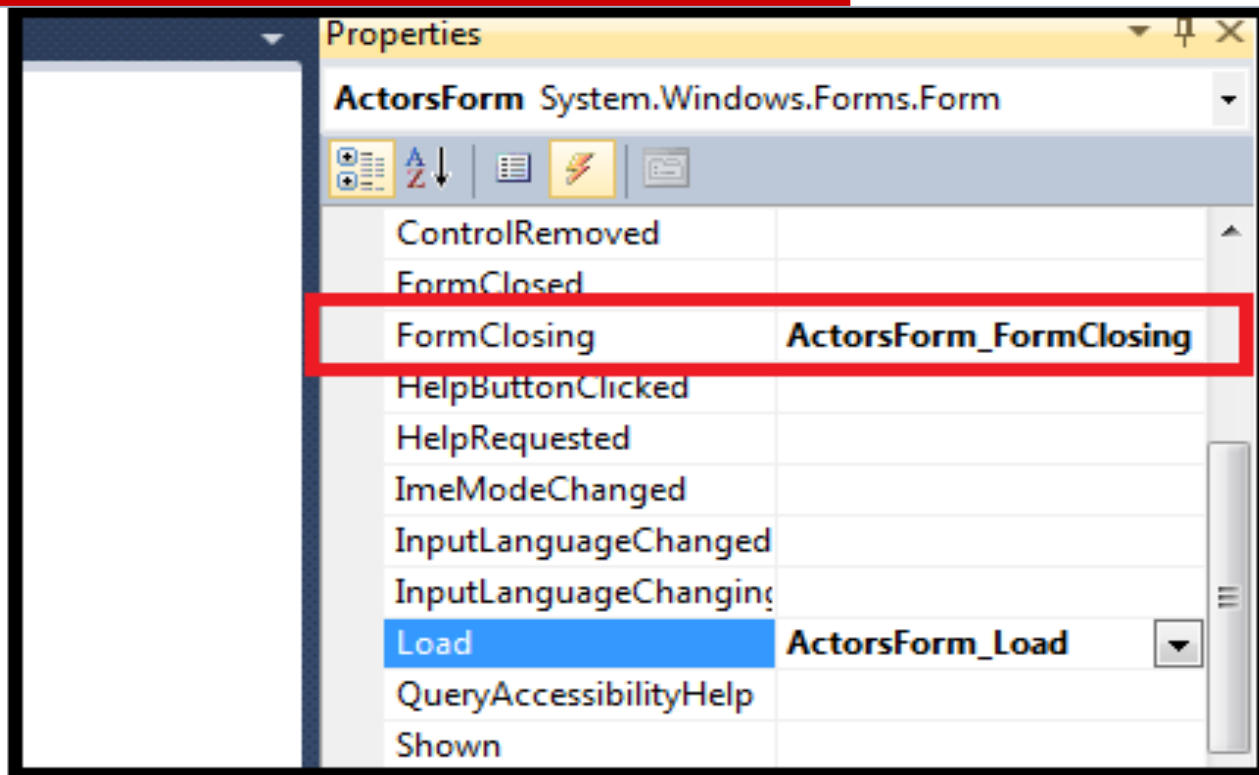
□ Close():

- This method closes the OracleDataReader and frees all resources associated with the OracleDataReader.

Closing the connection

- Either the connection object's **Close** or the **Dispose** method should be called to close the connection to the database. The Dispose method calls the Close method implicitly.
 - Free up system resources
 - More efficient application performance, which is especially important under high load conditions.

Closing the connection (Cont.)



```
private void ActorsForm_FormClosing(object sender, FormClosingEventArgs e)
{
    conn.Dispose();
}
```

Thank You