# Подготовка к работе

Лабораторные работы выполняются в среде MS Visual Studio 2010 и выше с учебной базой данных *AdventureWorksLT2008R2*, расположенной на MSSQL-2K8. Предварительно у преподавателя нужно получить логин и задать пароль для доступа к MS SQL Server, у вас доступ к этой базе – только чтение, поэтому Вам необходимо:

1. создать на сервере свою базу данных и
2. импортировать в нее таблицы из *AdventureWorksLT2008R2*: *Roles*, *Employees*, *SalesOrderHeader*, *SalesDetails*, *Product*, *ProductCategory, Customers,* затем
3. создать соответствующие ключи и связи между таблицами как показано на рис. 1-1 (все это можно выполнить в SQL Server Management Studio);

Далее в примерах кода вместо *AdventureWorksLT2008R2* указывайте имя своей БД, свои логин и пароль.

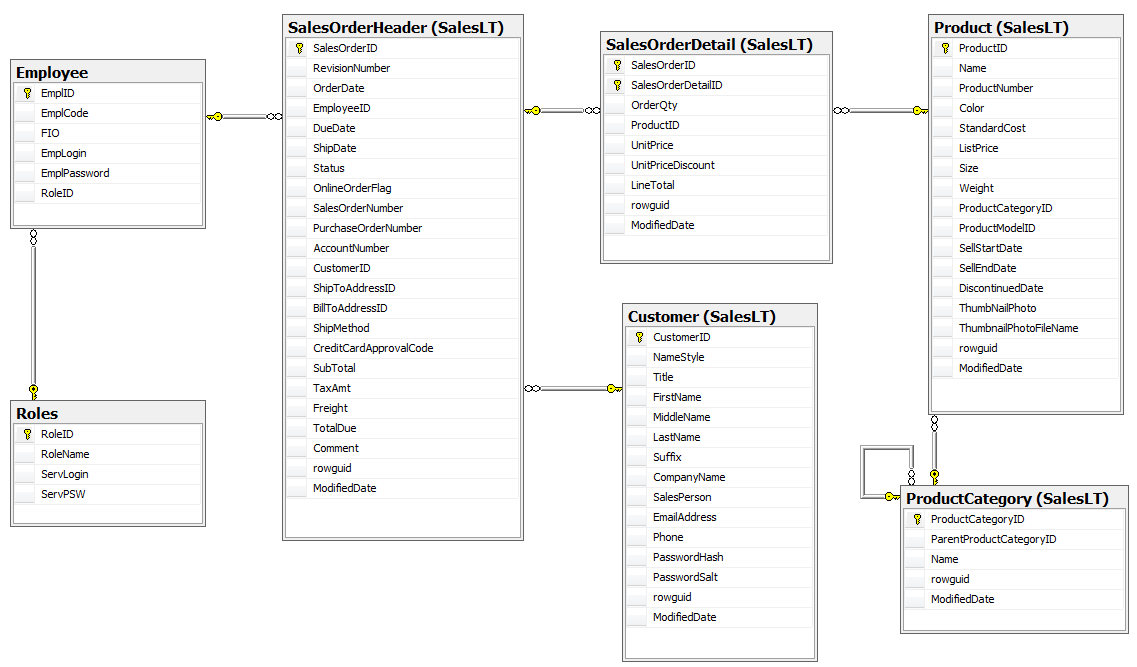


Рис. 1-1 Схема учебной базы данных

1. создайте WindowsFormsApplication в среде Visual Studio на языке C#; последовательность действий показана на рис. 1-2: ***Создать проект\Другие языки\Visual C#\Приложение Windows Forms*.**

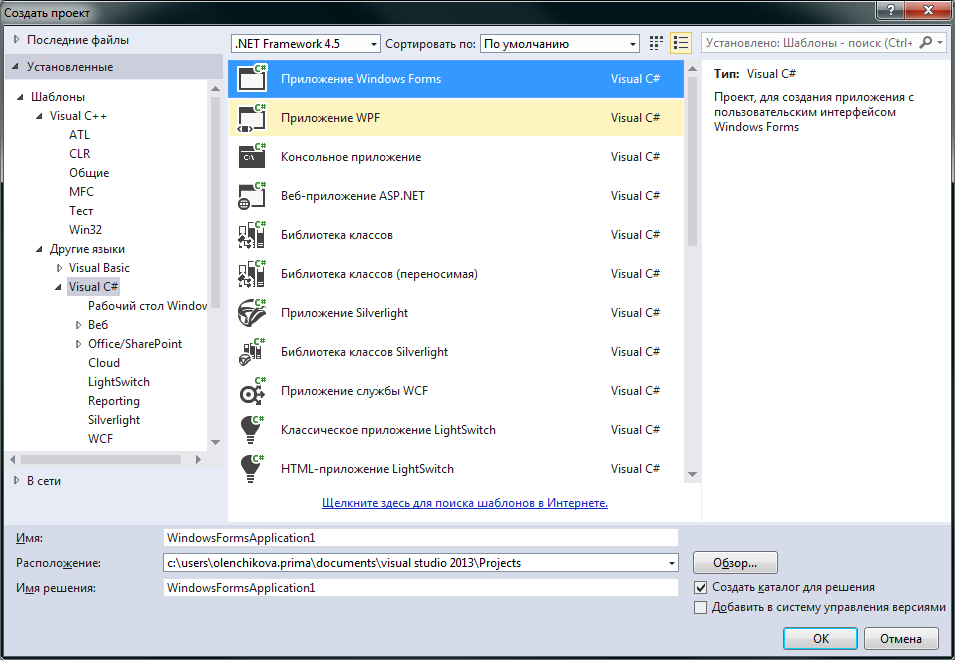


Рис. 1-2 Создание приложения для работы с базой данных

# Настройка подключений и подключение к данным

**Цель:** Научиться подключаться к БД из приложения, обрабатывать ошибки подключения. Ознакомиться с политикой безопасности при создании подключений к БД.

**Теоретический материал**

Перед началом работы с данными в приложении необходимо установить и открыть подключение и выбрать нужного поставщика данных. За все это отвечает объект *Connection*. Фактически объект Connection предоставляет канал для обмена командами и данными между СУБД и приложением. *Connection* имеет методы для работы с подключением (каналом), например, открывающие и закрывающие подключение, а также методы для работы с пулами подключений и транзакциями.

Среда Visual Studio предоставляет несколько способов создания и настройки подключения к БД: обозреватель серверов *Server Explorer*, мастер подключений *Data Source Configuration Wizard*, создание подключения *Connection* в коде.

*Server Explorer* используется для создания подключения на этапе разработки. Вы можете использовать Server Explorer для подключения к SQL Server и к другим БД из Visual Studio. Как только вы создали соединение с вашей БД, Server Explorer дает возможность просматривать ее схему, включая таблицы (и столбцы), представления, хранимые процедуры, данные и т. п. Работая с БД SQL Server, для изменения ее схемы также можно использовать Server Explorer. Для этой БД Server Explorer позволяет создавать, модифицировать или удалять таблицы, представления, хранимые процедуры и т. п. Нет необходимости создавать запросы (подобные CREATE TABLE) с помощью языка определения данных. Server Explorer дает возможность модифицировать схему посредством простого, но мощного пользовательского интерфейса – конструктора запросов QBE.

Мастер подключений *Data Source Configuration Wizard* предоставляет диалоговое средство автоматизации построения строк подключения. Строка подключения, созданная в результате запуска мастера, расположена в конфигурационном файле приложения. Сохранение своего подключения в конфигурационном файле полезно, если после развертывания вы хотите указать другой источник данных. Затем вы (или системный администратор) можете легко изменить конфигурационные настройки, вместо того чтобы изменять строку подключения в коде, компилировать и вновь развертывать приложение. Однако, если строка подключения содержит данные аутентификации, то ваше приложение становится уязвимым. Ниже мы рассмотрим, как можно создать безопасную строку подключения в конфигурационном файле.

Создание объекта *Connection* в программном коде обеспечивает наибольшую гибкость. Многие настройки соединения SQL Server можно задать только через строку подключения и только в коде. Для облегчения работы в коде со строками подключений существует конструктор строки подключения *SqlConnectionStringBuilder*. Конструкторы строк подключения упрощают процесс их построения, предоставляя множество часто используемых опций в качестве свойств, и избавляют программиста от необходимости помнить параметры строки подключения.

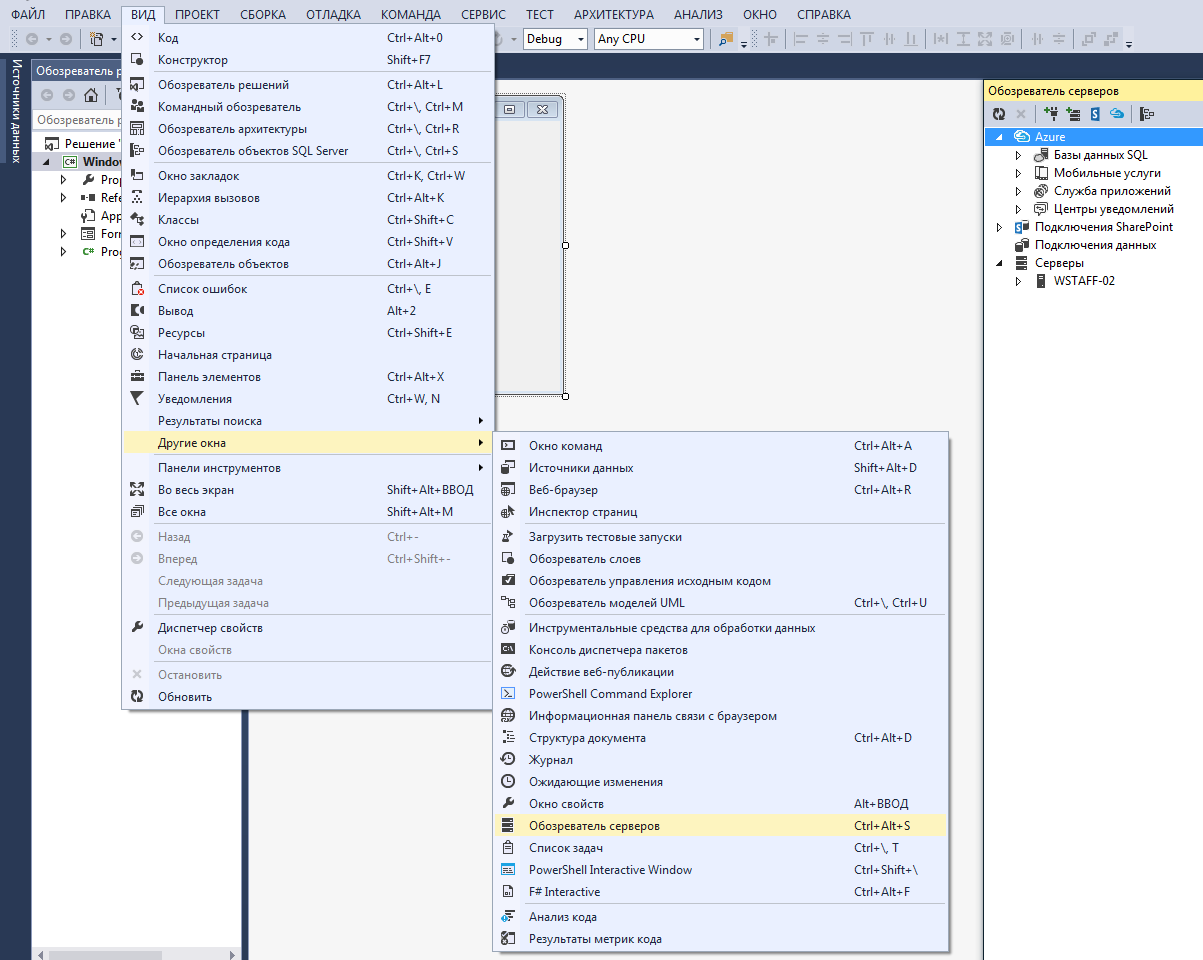
### Задание 1. Работа с БД с помощью *Server Explorer* (обозревателя серверов)

1. Если окно *Server Explorer* невидимо, в меню Вид (*View)* выберите, «обозреватель серверов» (Server Explorer) (нажмите сочетание клавиш *Ctrl+Alt+S* или пройдите путь ***Вид\Другие окна\Обозреватель серверов***, как показано на рис. 1-3).
2. Щелкните правой кнопкой мыши узел «Подключение данных» (*Data Connections)* и выберите *Add Connec­tion*. При первом добавлении подключения в Visual Studio появится диалоговое окно «Выбор источника данных» (*Choose Data Source)* (рис.1-4)*.*

ПРИМЕЧАНИЕ

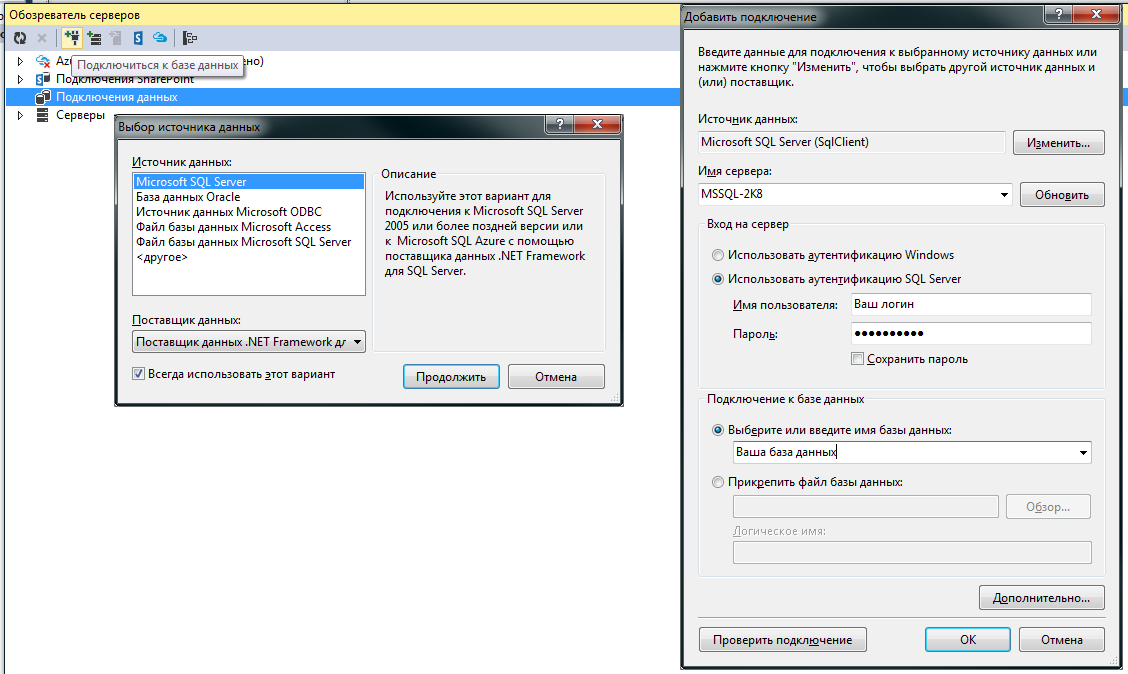
Если вместо диалогового окна Choose Data Source появится диалоговое окно Add Connection, щелкните кнопку Change, которая расположена в верхней части диа­логового окна Add Connection.

В диалоговом окне *Choose Data Source* (или аналогичном диалоговом окне *Change Data Source*) вы выбираете источник данных – *Microsoft Sql Server*, к которому хотите подключиться, а также провайдера данных для подключения (поставщика данных). Обратите внимание, как нужный провайдер данных автоматически заполняется при указании других источников данных. Вы можете выбрать любой допустимый провайдер для любого источника данных, но *Visual Studio* автоматически выбирает один из наиболее подходящих провай­деров данных. Поэтому поле *Поставщик данных* оставляем без изменения.



1)

Рис. 1-3 Подключение обозревателя серверов

**

2)

3)

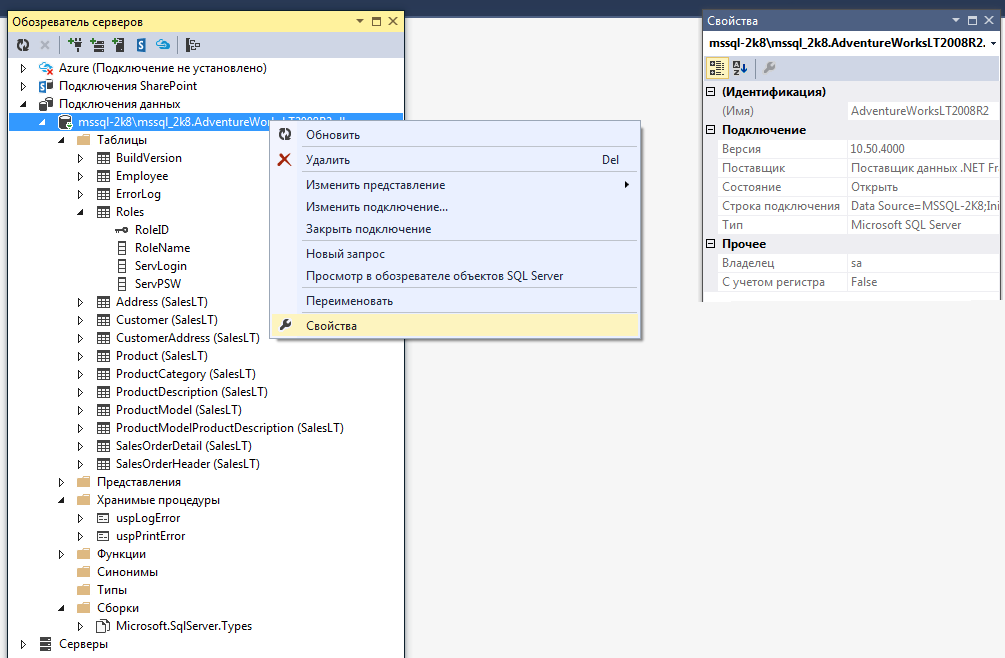
Рис. 1-4 Подключение к базе данных

1. В диалоговом окне *Добавить подключение* (или аналогичном диалоговом окне *AddConnection*) вы задаете:

* источник данных – Microsoft Sql Server (Sql Client),
* имя сервера – MSSQL-2K8,
* использовать аутентификацию SQL Server, Ваши логин и пароль на сервере,
* имя базы данных – Ваша база данных

Проверить, является ли подключение допустимым, можно, щелкнув *Проверить подключение*, а затем *ОК*, чтобы закрыть диалоговое окно и создать подключение в *Server Explorer*.

1. В окне *Server Explorer* появится узел с вашей базой данных. Раскрыв узел вы увидите объекты подключенной БД (рис. 1-5). Щелкните правой кнопкой мыши по имени БД и вы увидите свойства подключения.



4)

Рис. 1-5 Обозреватель серверов после подключения базы данных *AdventureWorksLT2008R2*

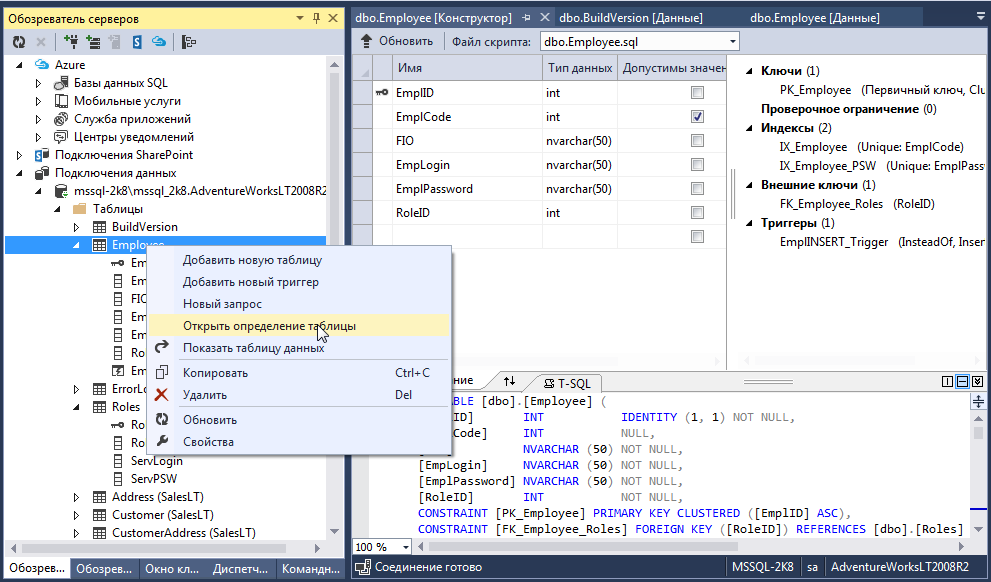
**Выполнение работы**

1. Для начала создайте подключение к образцу базы данных *AdventureWorksLT2008R2* (здесь и далее должна быть ваша база данных). Выполните шаги 1)-4)
2. После создания подключения в окне *Свойства* появится информация, связанная с подключением, а также информация о базе данных, к которой вы в действитель­ности подключены.
3. Выберите подключение, которое вы только что создали в окне Server Explorer, чтобы просмотреть доступную информацию в окне свойств.

ПРИМЕЧАНИЕ

Доступные свойства основаны на типе источника данных, к которому вы под­ключены, а также на состоянии подключения. Если подключение закрыто, вполне возможно, что вы увидите лишь небольшой список свойств, состоящий из строки подключения, которая применяется для подключения к базе данных, специального провайдера данных .NET Framework, используемого подключением, и состояния подключения. Чтобы просмотреть дополнительные свойства, необходимо открыть подключение, раскрыв узел подключения в Server Explorer. После открытия подклю­чение предоставляет дополнительные сведения, например, о владельце базы данных, типе и номере ее версии, а также о том, чувствительна ли база данных к регистру.

1. Для таблицы Employee посмотрите ее определение и определение объектов, связанных с ней. Щелкните правой кнопкой мыши по имени таблицы и выберите пункт Открыть определение таблицы, появится окно (рис. 1-6) с конструктором таблицы, SQL запросом на создание таблицы. **Проверьте и напишите в отчете, какие действия с таблицей можно выполнить в данном окне.**



4)

Рис. 1-6 Работа с таблицей БД

1. Создайте и выполните запрос к базе данных. Для этого щелкните правой клавишей на базу данных в «Обозревателе серверов», затем выберите пункт «Новый запрос», появится окно конструктора запросов (1-7).

**Самостоятельно создайте в базе данных новую таблицу Role\_1 с полями (роли пользователей БД, логин и пароль роли)**

**RoleID – int64, первичный ключ, автоинкремент;**

**RoleName – nchar(10), обязательное поле;**

**RoleLogin - nchar(20), обязательное поле;**

**RoleRassword – varchar(20) , обязательное поле.**

Если таблица Role\_1 успешно создана, она должна появиться в списке таблиц БД окна Обозревателя серверов. Если этого не произошло, попробуйте *обновить* отображение дерева таблиц.

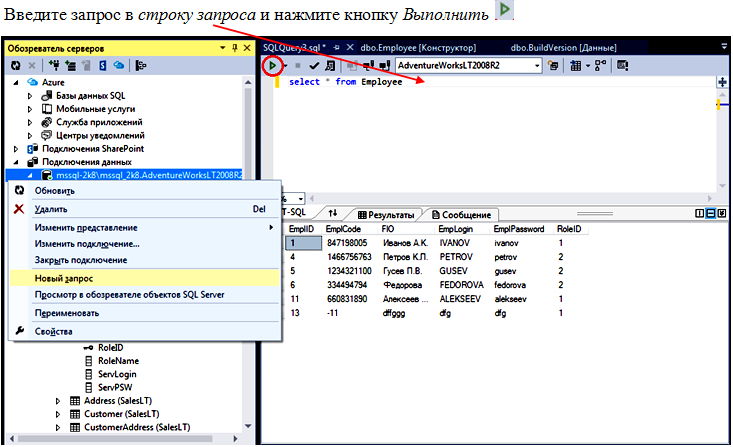


Рис. 1-7 Выполнение SQL запроса к БД

1. **Добавьте еще одну аналогичную таблицу Role\_2 , но теперь воспользуйтесь мастером создания таблиц.** Для вызова мастера щелкните правой кнопкой по узлу Таблицы и выберите пункт Добавить новую таблицу (рис. 1-8). Обратите внимание, что мастер предоставляет готовый шаблон для создания объектов, как в виде QBE-запроса, так и в виде SQL-запроса, писать можно как в одном, так и в другом окне.

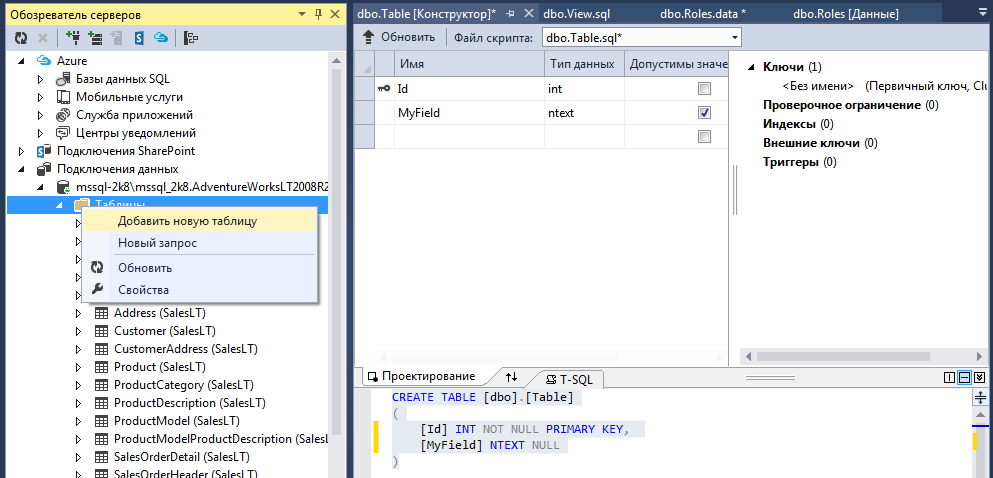


Рис. 1-8 Создание таблицы в БД

1. **Самостоятельно**, **проверьте, какие еще объекты БД можно создавать с помощью Server Explorer. Приведите скрины создания этих объектов**.

## Задание 2. Создание и настройка объекта *Connection* с помощью мастера *Data Source Configuration Wizard* мастера создания строк подключения

Для создания подключений к данным с помощью мастера *Data Source Configuration Wizard* выполните следующее.

1. Выберите *Добавить новый источник данных*  в меню *ПРОЕКТ* (рис. 1-9) либо на вкладке *Источник данных* нажмите пиктограмму 

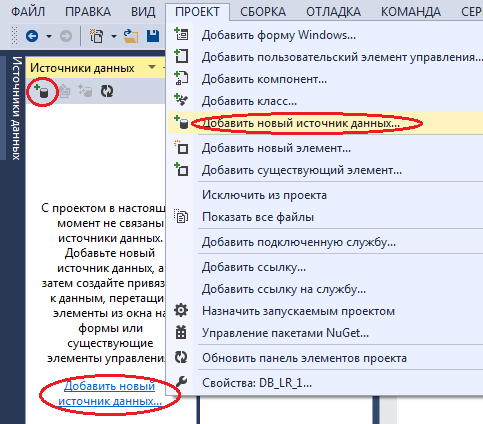


Рис. 1-9 Запуск мастера *Data Source Configuration Wizard*

1. Тип источника данных по умолчанию — База данных, поэтому просто щелкните *Далее* и еще раз *Далее* (рис. 1-10)

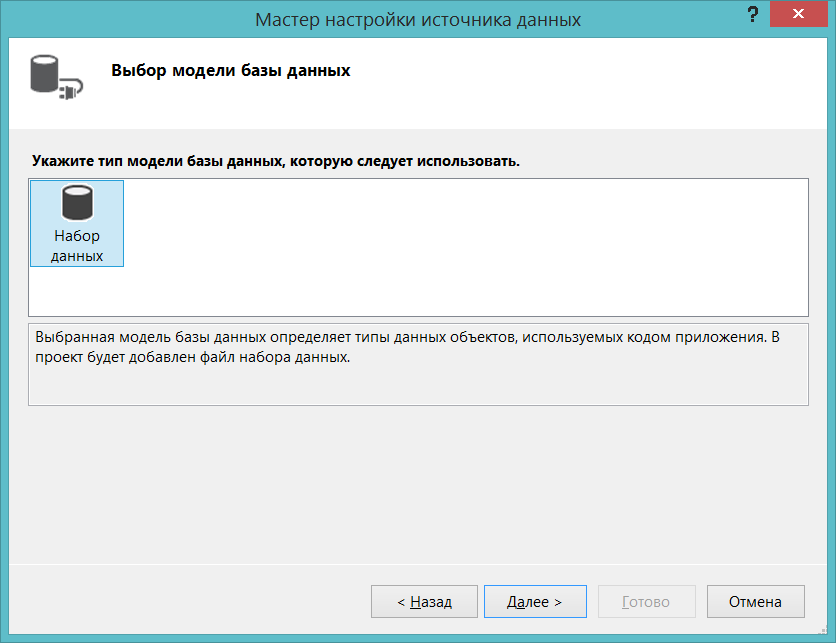
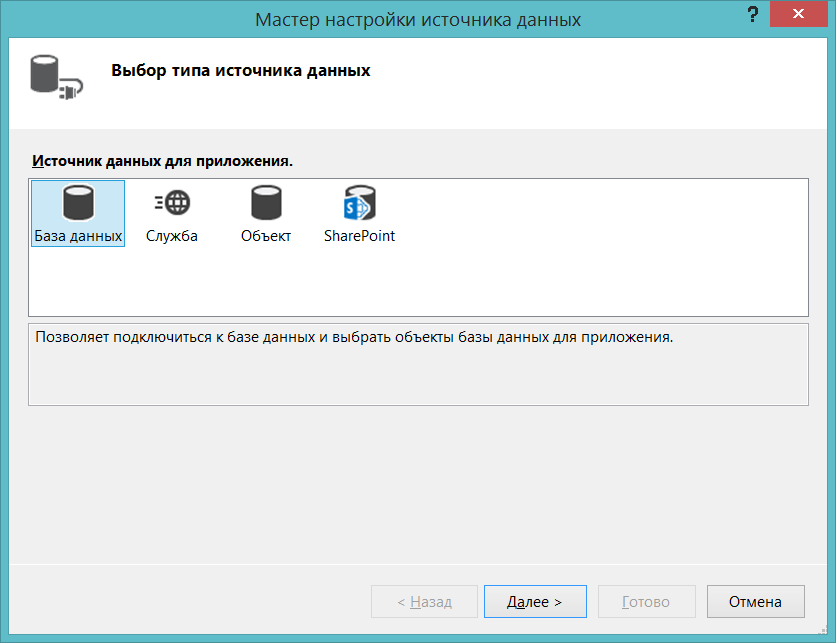


Рис. 1-10 Настройка источника данных

1. Создайте объект Connection на странице *Выбор подключения базы данных* мастера. Для этого упражнения мы создадим новое подключение, поэтому щелкните кнопку *Создать подключение*, чтобы открыть диалоговое окно Add Connection (рис.1-11). Далее действуйте, как в задании1: введите в область имени сервера имя своего SQL Server; выберите соответствующий метод аутентификации для получения доступа к свое­му SQL Server; укажите имя вашей базы данных; проверьте, является ли подключение допустимым, щелкнув «Проверить подключение» (Test Connection), после чего закройте диалоговое окно кнопкой ОК.

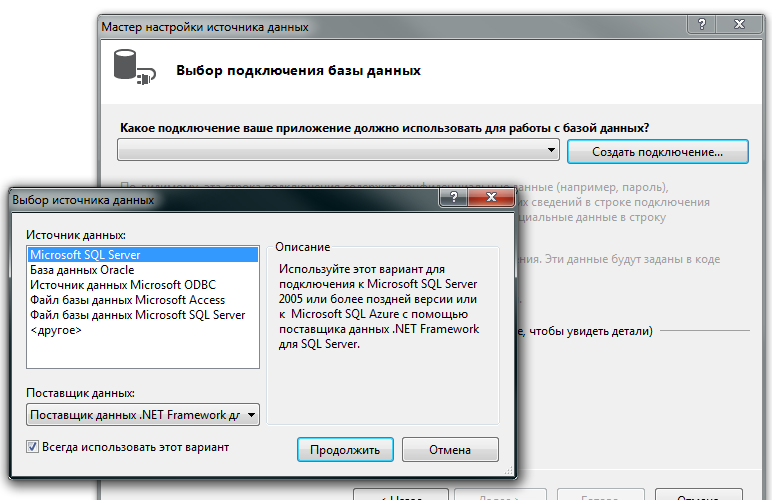


Рис. 1-11 Создание подключения к базе

ПРИМЕЧАНИЕ

Если ваше подключение использует аутентификацию SQL и требует имя пользова­теля и пароль для подключения к базе данных, значит, опция, предназначенная для включения или исключения уязвимых данных в строку подключения, активна. Стро­ка подключения не содержит уязвимых данных по умолчанию. При необходимости открыть подключение вам придется предоставить эту информацию приложению и подключиться к базе данных. Вы можете выбрать опцию, чтобы отобразить уязвимые данные в строке подключения, но лучше этого не делать, поскольку пользователи, у которых есть доступ к строке подключения, смогут увидеть пароль. Лучше вос­пользоваться опцией интегрированной безопасности.

На данном этапе вы успешно создали подключение к данным и можете просматри­вать строку подключения, раскрыв узел *Строка подключения…* (рис. 1-12)

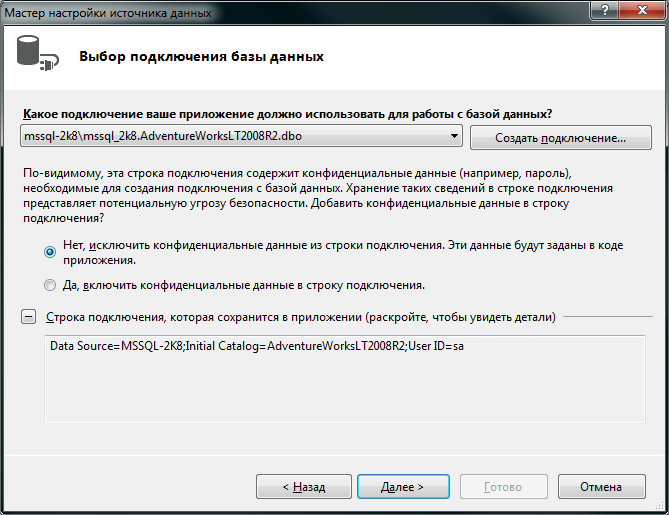


Рис.1-12 Сформированная строка подключения

Убедитесь, что установлен флажок *Нет, исключить конфиденциальные данные из строки подключения*. **Самостоятельно, что будет, если установить флажок *Да, включить конфиденциальные данные из строки подключения.***

Далее следует добавить подключение к проекту, выполняя следующие шаги мастера.

1. Щелкните *Далее*. Вам предлагается сохранить строку Connection в конфигураци­онном файле приложения, а также дать имя подключению (рис. 1-13). Сохранение своего подключения в конфигурационном файле полезно, если после развертывания вы хотите иметь возможность указать другой источник данных. Затем вы (или системный администратор) можете легко изменить конфигурационные настройки, вместо того чтобы изменять строку подключения в коде, компилировать и вновь развертывать приложение.

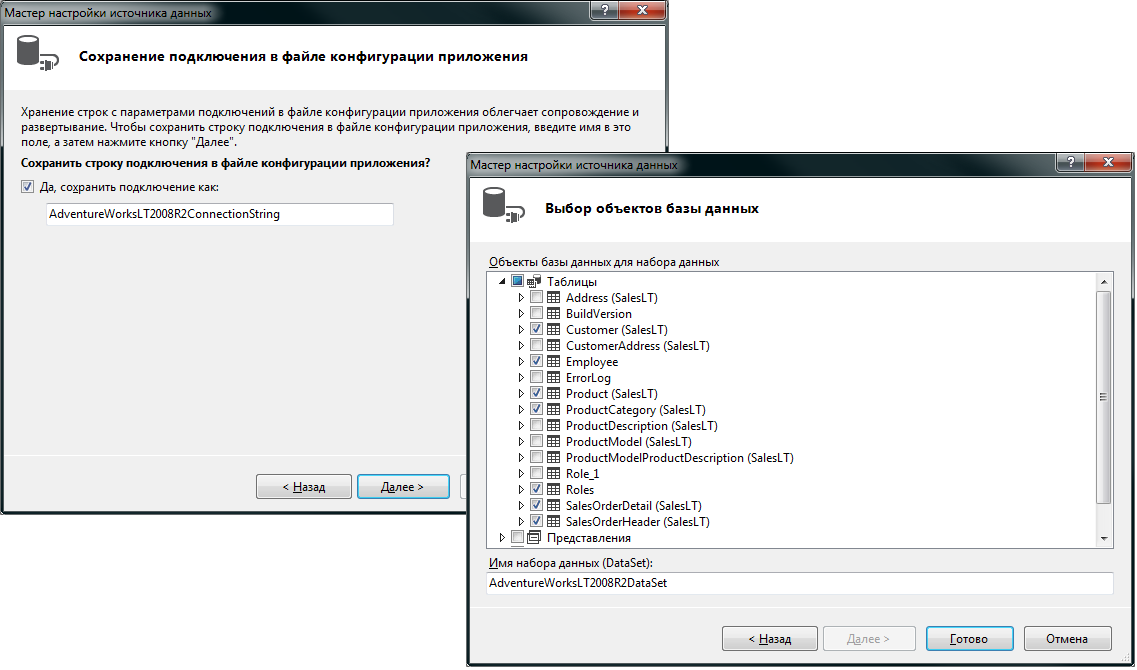


Рис. 1-13 Выбор таблиц, и других объектов, для подключения

Ког­да строка подключения сохранена в конфигурационном файле приложения, к ней можно получать доступ и изменять с помощью Project Designer. Откройте Project Designer, дважды щелкнув мышью изображение Properties в Solution Explorer. Чтобы получить доступ к строкам подключения, которые хранятся в вашем приложении, щелкните вкладку Settings.

1. Страница *Выбор объектов базы данных* мастера позволяет выбирать *Таблицы*, *Представления*, *Хранимые процедуры*  и т. д. для работы в приложении. В данном случае раскройте узел *Таблицы* и выберите таблицы *Customer, Employee, Product, ProductCategory, Roles, SalesOrderDetail* и *SalesOrderHeader*. Щелкните *Готово*.
2. На данном этапе вы создали в проекте *Типизированный набор данных.*

Чтобы увидеть добавленные объекты выберите в меню пункты «Данные», «Показать источники данных» (рис.1-14). В окне *Обозреватель решений*, созданный набор данных *xxxDataSet.xsd*, в окне Источники данных – объекты базы данных, а, раскрыв узел *App.config* – строку подключения и ее имя в проекте.

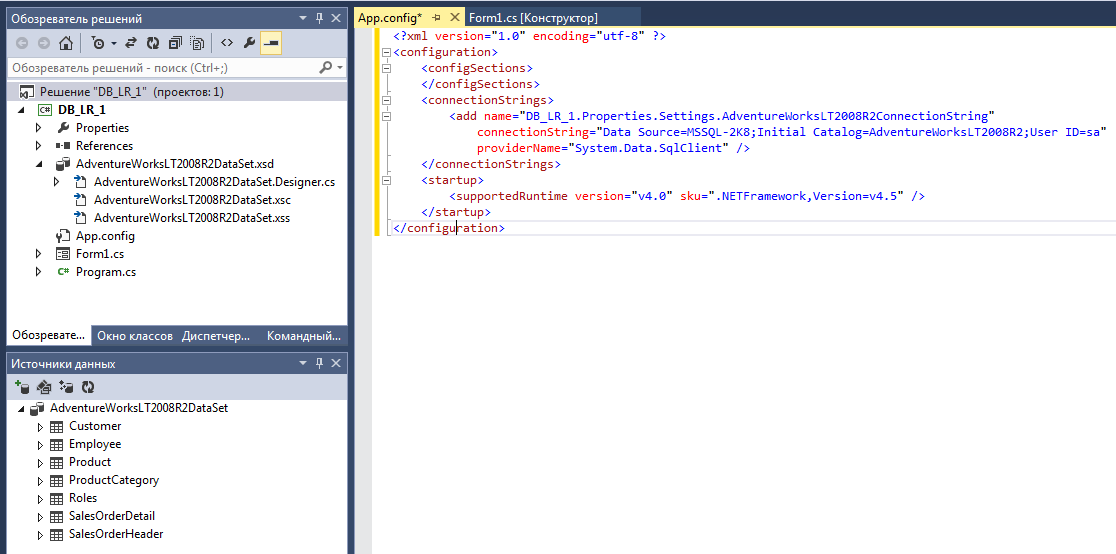
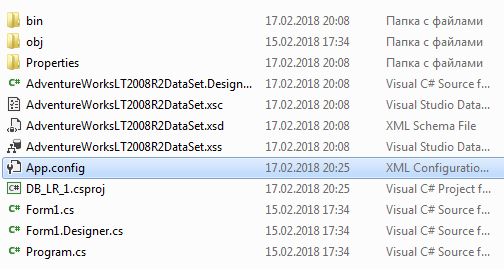
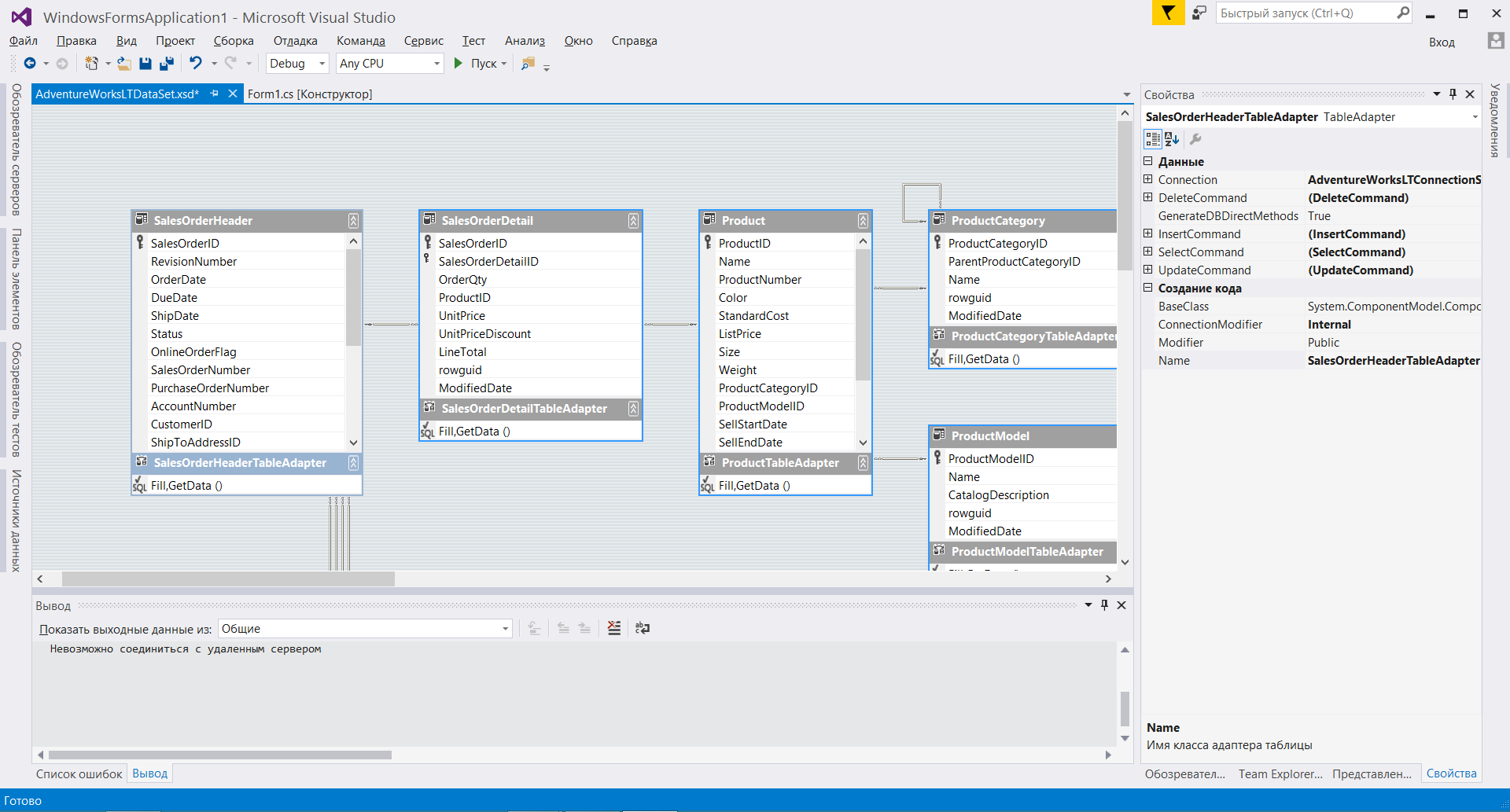
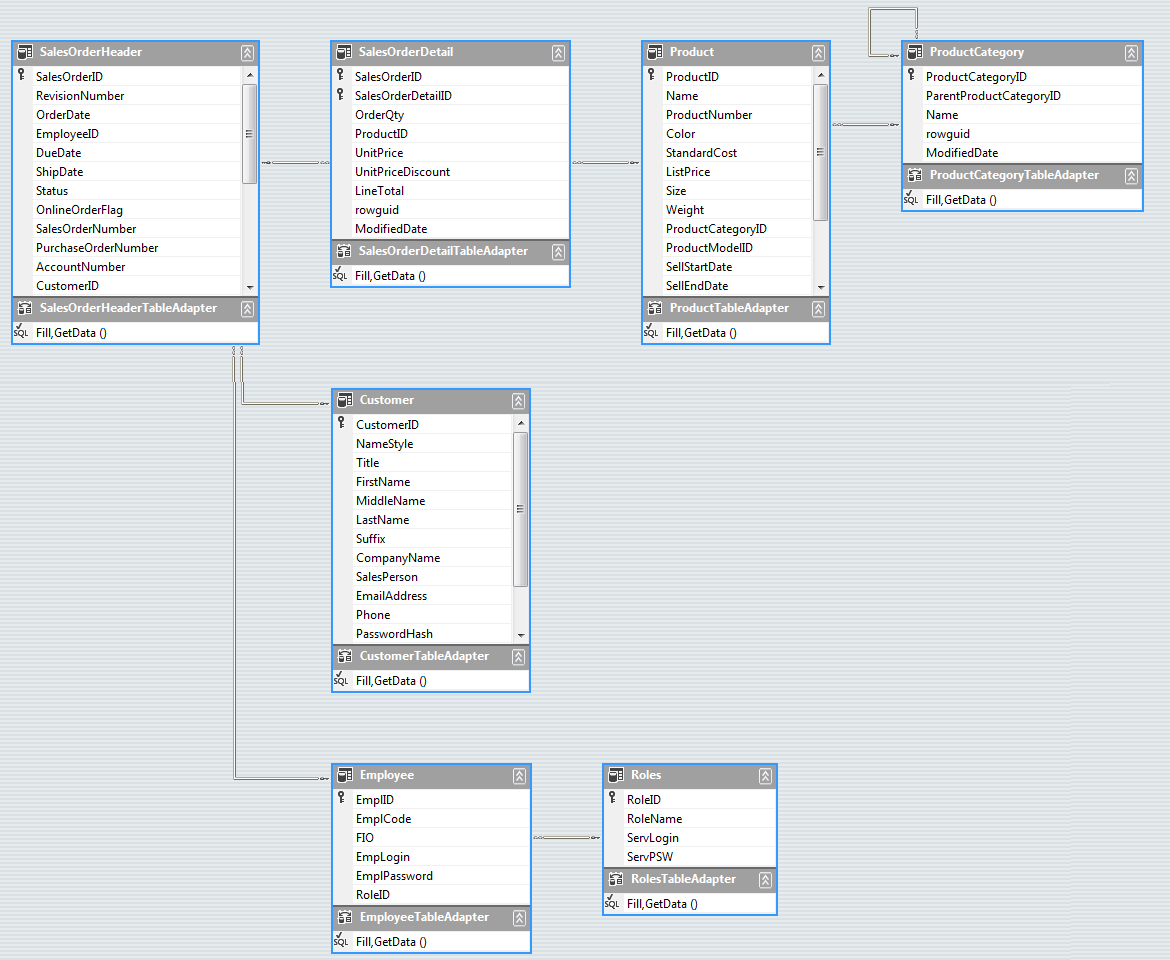


Рис. 1-14 Объекты, созданные мастером подключений *Data Source Configuration Wizard*

1. Теперь, когда вы закончили работать с мастером, посмотрите, где находится под­ключение и что оно содержит. Подключение, созданное в результате запуска мастера, расположено в сгенерированном конструктором файле с кодом набора данных. Чтобы просмотреть текущий объект Connection, откройте набор данных в Dataset Designer, дважды щелкнув объект Dataset в «Обозревателе решений» (Solution Explorer). (Объектом Dataset является узел AdventureWorksLT2008R2DataSet.xsd.) Выберите строку заголовка TableAdapter в конструкторе (на­пример, CustomersTableAdapter). Информация о подключении доступна в окне Prop­erties, где можно раскрыть узел и увидеть имя, модификатор и строку подключения. Поинтересуйтесь какие команды созданы в InsertCommand, DeleteCommand, SelectCommand, UpdateCommand, раскрыв соответствующий узел.
2. 



ПРИМЕЧАНИЕ

Свойство ConnectionString отображает строку подключения, сохраненную в конфи­гурационном файле приложения. Изменение строки подключения в окне Properties подобно редактированию строки подключения в файле Settings и влияет на все подключения, которые ссылаются на эти настройки.

## Задание 3. Создание объектов *Connection* SQL Server в коде

Чтобы программно создать подключение с использованием четырех основных провайдеров данных, необходимо начать с создания экземпляра нового объекта *Con­nection* с помощью ключевого слова *New* и установки его свойства *ConnectionString*. Вы можете создать подключение и задать строку подключения в одном конструкторе или же привязать строку подключения к свойству *SqlConnection.ConnectionString* после создания *Con­nection*. Убедитесь, что заменили *ServerName* и *Initial Catalog* допустимыми вашей средой значениями. Чтобы исключить необходимость уточнять полное имя объектов в коде, добавьте выражение *using System. Data.SqlClient* в начало файла с кодом. Если вашему приложению необходимо отреагировать на события объектов *Connection*, создайте обработчики события в С#.

ПРИМЕЧАНИЕ

По умолчанию приложения Microsoft Windows создаются в Visual Studio со ссылками на пространства имен *System.Data.SqlClient, System.Data.OleDb* и *System.Data.Odbc,* так что они тут же становятся доступными для использования и появляются в IntelliSense без каких-либо дальнейших действий. По умолчанию ссылка на пространство имен *SystemData.OracleClient* не включена. Ее необходимо добавить в приложение для создания объектов *OracleConnection.*

**Создание объектов SqlConnection:**

SqlConnection ConnToSql = new SqlConnection(@"Data Source=ServerName;”+

"Initial Catalog=DatabaseName;Integrated Security=False; ”+

”UserID=ваш логин; Password=ваш пароль");

ДОПОЛНИТЕЛЬНО

**Создание объектов Connection ODBC в коде (подключение к БД Nwind Access)**

Добавьте в начало кода выражение using System.Data.Odbc;

OdbcConnection ConnToOdbc = new OdbcConnection("Dsn=MS Access Database; " +

" dbq=C:\\DataSources;driverid=281; fil=MS Access;maxbuffersize=2048; " +

" pagetimeout=5; uid=admin");

Убедитесь, что заменили строку подключения допустимой строкой для источника дан­ных ODBC, к которому необходимо подключиться.

**Создание объектов Connection OLE DB в коде (подключение к БД Nwind Access)**

Добавьте в начало кода выражение using System.Data.OleDb;

OleDbConnection ConnToOleDb = new 0leDbConnection("Provider=" + "Microsoft.Jet.OLEDB.4.0;Data Source=\"Nwind.mdb\";" +

"Persist Security Info=False");

Чтобы научиться работать с объектами Connection, выполните следующее.

1. Создайте новое приложение Windows и назовите его DataConnections.
2. Добавьте в форму девять кнопок, устанавливая свойства Name и Text, как показано в табл. 1-1.

**Табл. 1-1.** Настройки кнопок формы Data Connections

|  |  |
| --- | --- |
| Свойство *Name* | Свойство *Text* |
| *OpenSqlButton* | Open SQL |
| *OpenOleDbButton* | Open OLE DB |
| *OpenOdbcButton* | Open ODBC |
| *CloseSqlButton* | Close SQL |
| *CloseOleDbButton* | Close OLE DB |
| *CloseOdbcButton* | Close ODBC |
| *GetSqllnfoButton* | Get SQL Info |
| *GetOleDblnfoButton* | Get OLE DB Info |
| *GetOdbcInfoButton* | Get ODBC Info |

1. Добавьте в форму три Label, установив свойства Name и Text, как показано в табл. 1-2.

**Табл. 1-2**. Настройки надписей формы Data Connections

|  |  |  |  |
| --- | --- | --- | --- |
| Свойство Name | Свойство Text | | |
| *SqlConnectionStateLabel* | | Closed |
| *OleDbConnectionStateLabel* | | Closed |
| *OdbcCormectionStateLabel* | | Closed |
|  | |  |

Расположите элементы управления, чтобы форма выглядела, как на рис. 1-2.

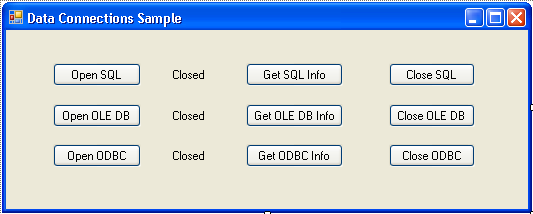


Рис. 1-2

1. Откройте код только что созданной вами формы.
2. Добавьте код для создания всех трех объектов *Connection*. Код должен выглядеть следующим образом

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Data.OleDb;

using System.Data.Odbc;

namespace DataConnections

{

public partial class Forml : Form

{

public Forml()

{

InitializeComponent();

}

// Объявление объектов Connection для трех провайдеров (вместо красного текста поставьте ваши данные)

private SqlConnection ConnectionToSql = new SqlConnection(

"Data Source=MSSQL-2K8;Initial Catalog= AdventureWorksLT2008R2;" +

"Integrated Security=False; User ID=логин; Password=пароль");

private OleDbConnection ConnectionToOleDb = new

System.Data.OleDb.OleDbConnection("Provider=" +

"Microsoft.Jet.OLEDB.4.0;Data Source=H:\\Учебные материалы\\" +

"4 курс\\DATABASE\\Лабораторные работы ADO.NET\\Борей.mdb");

private OdbcConnection ConnectionToOdbc = new OdbcConnection("Dsn=" +

"MS Access Database;dbq=H:\\Учебные материалы\\" +

"4 курс\\DATABASE\\Лабораторные работы ADO.NET\\Борей.mdb;" +

"defaultdir=H:\\Учебные материалы\\4 курс\\DATABASE\\" +

"Лабораторные работы ADO.NET;driverid=281; " +

"fil=MS Access;maxbuffersize=2048;pagetimeout=5;uid=admin");

}

}

Для открытия подключений к базе данных воспользуйтесь методом *Open* объекта *Connection*. Чтобы это продемонстрировать, необходимо вызвать метод *Open* для каждого подключения щелчком кнопки *Open*.

1. Создайте обработчики события для кнопок Open всех провайдеров и добавьте представленный ниже код, который будет открывать подключение к базе данных с помощью этих кнопок.

private void OpenSqlServerButton\_Click(object sender, EventArgs e)

{

ConnectionToSql.Open();

}

private void OpenOleDbButton Click(object sender, EventArgs e)

{

ConnectionToOleDb.Open();

}

private void OpenOdbcButton\_Click(object sender, EventArgs e)

{

ConnectionToOdbc.Open();

}

Чтобы закрыть подключение к базе данных, воспользуйтесь методом Close объекта Connection. Сле­дует отметить, что вызов метода Close откатывает все отложенные транзакции и освобождает подключение, так что оно возвращается в пул подключений. Чтобы все это выполнить, создайте обработчики события для кнопок Close всех провай­деров и добавьте код, чтобы вызвать метод Close для тела обработчика.

1. Добавьте методы Close в обработчики события, чтобы закрыть подключение к базе данных, щелкая кнопки Close.

private void CloseSqlButton\_Click(object sender, EventArgs e)

{

ConnectionToSql.Close();

}

private void Close01eDbButton\_Click(object sender, EventArgs e)

{

ConnectionToOleDb.Close();

}

private void CloseOdbcButton\_Click(object sender, EventArgs e)

{

ConnectionToOdbc.Close();

}

Когда состояние подключения изменяется, значение свойства *CurrentState* объекта Connection обновляется, чтобы отобразить текущее состояние подключения. При открытии и закрытии подключения можно проверить значение этого свойства и убедиться, что подключение действительно открывается или закрывается. Все подключения вызывают событие *StateChange*, на которое можно ответить, чтобы контролировать состояние подключения. Для заполнения надписи о состоянии подключения необходимо создать обработчики события *StateChange* всех провай­деров, добавить код, обновляющий надписи о состоянии подключения вместе со значением свойства *CurrentState* объекта *Connection*, которое предоставляется как аргумент события.

1. Добавьте в форму представленный ниже код, обновляющий значения надписей о состоянии подключения всякий раз, когда изменяется текущее состояние под­ключения. Создайте обработчик загрузки формы для C# для того, чтобы можно было добавить обработчики события *StateChange*.

private void Form1\_Load(object sender, EventArgs e)

{

ConnectionToSql.StateChange += new System.Data.StateChangeEventHandler

(this.ConnectionToSql\_StateChange);

ConnectionToOleDb.StateChange += new System.Data.StateChangeEventHandler

(this.ConnectionToOleDb\_StateChange);

ConnectionToOdbc.StateChange += new System.Data.StateChangeEventHandler

(this.ConnectionToOdbc\_StateChange);

}

private void ConnectionToSql\_StateChange(object sender, StateChangeEventArgs e)

{

SqlConnectionStateLabel.Text = e.CurrentState.ToString();

}

private void ConnectionToOleDb\_StateChange(object sender,

StateChangeEventArgs e)

{

OleDbConnectionStateLabel.Text = e.CurrentState.ToString();

}

private void ConnectionToOdbc\_StateChange(object sender,

StateChangeEventArgs e)

{

OdbcConnectionStateLabel.Text = e.CurrentState.ToString();

}

1. Нажмите F5, чтобы запустить приложение, протестировать форму и посмотреть, как работает функциональность.
2. Когда откроется форма, щелкните кнопку Open SQL и проверьте, изменяется ли надпись о состоянии подключения, информирующая о том, что подключение сейчас открыто.
3. Щелкните кнопку Open SQL и проверьте, изменяется ли надпись о состоянии подключения, отображающая текущее состояние подключения, которое на данный момент закрыто.

Чтобы продемонстрировать использование события *InfoMessage*, необходимо со­здать обработчик события для обработки сообщения. Чтобы исключить необходи­мость создания объекта базы данных, генерирующего негрубую ошибку, мы вос­пользуемся встроенной в объект *SqlConnection* возможностью, которая позволяет перехватывать ошибки с уровнем значимости до 16, задав свойству *FirelnfoMessageEventOnUserErrors* объекта Connection значение True до выполнения метода, который выдаст Ошибку.

1. Добавьте следующий код, который будет обрабатывать событие Click кнопки GetSqllnfo и событие *InfoMessage* объекта *SqlConnection*. При просмотре кода в событии Click кнопки вы увидите, что мы собираемся из­менить имя базы данных, с которой установлено подключение, на неверное, в ре­зультате чего будет сгенерирована ошибка с уровнем значимости 11 и произойдет событие InfoMessage. После этого код в обработчике события InfoMessage откроет окно сообщения, где будет отображаться ошибка.

// Добавьте эту строку кода в форму, чтобы подключить обработчик

// события InfoMessage

ConnectionToSql.InfoMessage += new

System.Data.SqlClient.SqlInfoMessageEventHandler

(this.ConnectionToSql\_InfoMessage);

private void GetSqlInfoButton\_Click(object sender, EventArgs e)

{

ConnectionToSql.FireInfoMessageEventOnUserErrors=true;

ConnectionToSql.ChangeDatabase("AdventureWorksLT2008R2l");

}

private void ConnectionToSql\_InfoMessage(object sender,

SqlInfoMessageEventArgs e)

{

MessageBox.Show(e.Message);

}

Помимо предыдущих типов информации, доступных благодаря объектам Connec­tion, можно также получить некоторые метаданные из источника данных, к которо­му вы подключены. Вами были изучены свойства объекта Connection в окне Properties всех доступных в Server Explorer подключений. Во время выпол­нения эта информация доступна также благодаря объекту Connection. В качестве примера добавьте в приложение еще несколько строк кода и реализуйте кнопки Get Info оставшихся подключений, чтобы получить версии сервера источников данных, к которым они подключены.

1. Добавьте в форму следующий код:

private void GetOleDbInfoButton\_Click(object sender, EventArgs e)

{

MessageBox.Show(ConnectionToOleDb.ServerVersion.ToString(),

"Server Version");

}

private void GetOdbcInfoButton\_Click(object sender, EventArgs e)

{

MessageBox.Show(ConnectionToOdbc.ServerVersion.ToString(),

"Server Version");

}

Теперь запустите приложение еще раз, чтобы проверить наличие дополнительной функциональности и убедиться, что объекты Connection предоставляют информа­ционное сообщение и метаданные.

1. Нажмите клавишу F5 для запуска приложения.
2. Щелкните кнопку Open SQL, чтобы открыть подключение к SQL Server и обновить надпись о состоянии подключения.
3. Щелкните кнопку Get SQL Info, чтобы изменить базу данных на неверную Northwind и сгенерировать событие InfoMessage, которое будет отображено в окне сообщения.
4. Щелкните кнопку Get OLE DB Info, чтобы получить информацию о версии сервера источника данных OLE DB.
5. Щелкните кнопку Close OLE DB, чтобы закрыть подключение и обновить надпись о его состоянии.
6. Сохраните приложение.

## Задание 4. Обработка ошибок подключения к базе данных

Когда SQL Server возвращает предупреждение или ошибку, провайдер данных .NET Framework для SQL Server создает *SqlException*, которое можно перехватить в приложении, чтобы решить проблему. Когда генерируются *SqlException*, проверьте свойство *SqlException.Errors*, чтобы получить доступ к коллекции ошибок, которые возвращаются из сервера SQL. Свойство SqlException.Errors — это класс SqlErrorCollection (коллекция классов SqlError), который всегда содержит как минимум один объект SqlError.

1. Создайте новое приложение Windows и назовите его HandlingConnectionErrors.
2. Добавьте в форму 3 кнопки и установите следующие свойства.

Button 1:

1. Name - GoodConnectButton;
2. Text - Connect (допустимая строка подключения).

Button2:

1. Name - ConnectToInvalidUserButton;
2. Text - Connect to invalid user.

Button3:

1. Name - ConnectToInvalidDatabaseButton;
2. Text - Connect to invalid database.
3. Дважды щелкните каждую кнопку, чтобы создать обработчики события Click кноп­ки, и откройте код.
4. Добавьте выражение для пространства имен using System.Data.SqlClient*.*
5. Следующий код создает новое подключение, основанное на строке подключения, которая была ему передана, пытается открыть подключение, а затем выводит на экран любые произошедшие ошибки. Добавьте этот код под обработчиками собы­тия Click кнопки:

private void ConnectToDatabase(string connectionstring)

{

SqlConnection connection = new SqlConnection(connectionString);

try

{

connection. Open();

}

catch (SqlException ex)

{

string errorMessage = "";

// Итерация всех возвращаемых ошибок,

foreach (SqlError ConnectionError in ex.Errors)

{

errorMessage += CcnnectionError.Message + " (error: " +

ConnectionError.Number.ToString() + ')" + Environment.Newline;

if (ConnectionError.Number == 18452)

MessageBox.Show("Invalid Login Detected, please provide valid"+

"credentials!");

}

MessageBox.Show(errorMessage);

}

finally

{

connection.Close();

}

}

1. Добавьте код так, чтобы обработчики события Click трех кнопок выглядели сле­дующим образом:

private void GoodConnectButton\_Click(object sender, EventArgs e)

{

// Это правильная строка подключения (Исправьте для своей БД)

String GoodConnection = "Data Source=.\\sqlexpress;Initial Catalog = "+

"AdventureWorksLT2008R2;Integrated Security=True";

ConnectToDatabase(GoodConnection);

}

private void ConnectToInvalidUserButton Click(object sender, EventArgs e)

{

// Эта строка подключения имеет неправильный мандат String

InvalidUserConnection = "Data Source=" +

".\\sqlexpress;Initial Catalog=AdventureWorksLT2008R2;User ID = InvalidUser”;

ConnectToDatabase(InvalidllserConnection);

}

private void ConnectToInvalidDatabaseButton\_Click(object sender,

EventArgs e)

{

// Это неправильная строка подключения

String InvalidDatabaseConnection = "Data Source=" +

".\\sqlexpress;Initial Catalog=InvalidDatabase;Integrated" +

"Security=True";

ConnectToDatabase(InvalidDatabaseConnection);

}

1. Запустите приложение.
2. Щелкните кнопку Connect (допустимая строка подключения). Не должно быть сгенерировано ни одной ошибки.
3. Щелкните кнопку Connect То Invalid User, чтобы был выполнен код для перехвата особой ошибки при входе в систему (error 18452).
4. Щелкните кнопку Connect То Invalid Database — и вы увидите, что ошибка была сгенерирована и отображена в MessageBox.

## Задание 5. Возвращение списка доступных *SQL Server* в сети

.NET Framework предоставляет приложению возможность нахождения экземпля­ров SQL Server в сети, так что ваши программы при необходимости могут обрабаты­вать эту информацию. Чтобы получить список доступных экземпляров SQL Server, воспользуйтесь свойством Instance класса SqlDataSourceEnumerator и вызовите метод GetDataSources. Данный метод возвращает DataTable, который содержит информацию обо всех видимых в сети сервервх SQL. Возвращенный DataTable содержит столбцы, приведенные в табл. 1-3.

**Табл. 1-3.** Схема DataTable, возвращенного методом *GetDataSources*

|  |  |
| --- | --- |
| **Столбец** | **Описание** |
| *ServerName*  *InstanceName*  *IsClustered*  *Version* | Имя сервера SQL, который содержит видимый экземпляр  Имя экземпляра сервера или пустой рабочий экземпляр по умолчанию Указывает, является ли сервер частью кластера  Номер версии SQL Server |

Чтобы иметь представление, как можно получить список видимых SQL Server, в этом упражнении вы создадите небольшое приложение для отображения информации, полученной при помощи метода GetDataSources DataGridView.

1. Создайте новое приложение Windows под названием SqlServerEnumerator.
2. Добавьте в форму DataGridView и назовите VisibleSqlServers.
3. Добавьте элемент управления Button под элемент управления Grid и установите его свойству Name значение GetDataSourcesButton.
4. Задайте свойству Text кнопки значение Get Visible Servers.
5. Дважды щелкните кнопку Get Visible Servers для создания обработчика события Click и откройте код.
6. Добавьте код, чтобы обработчик имел следующий вид:

System.Data.Sql.SqlDataSourceEnumerator instance =

System.Data.Sql.SqlDataSourceEnumerator.Instance;

VisibleSqlServers.DataSource = instance.GetDataSources();

1. Теперь запустите приложение и щелкните кнопку Get Visible Servers. В Grid явятся все видимые SQL Server сети, как на рис. 1-3

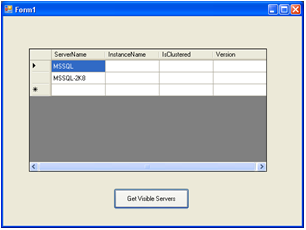


Рис. 1-3

## Задание 6. Защита уязвимых данных строки подключения с помощью шифрования

Поскольку большинство данных в реальных сценариях уязвимы, очень важным явля­ется вопрос защиты серверов и баз данных от несанкционированного доступа. Чтобы обеспечить ограниченный доступ к своему источнику данных, лучше всего защитить такую информацию, как идентификаторы пользователей, названия источников данных и, конечно же, пароли. Не рекомендуется хранить подобную информацию в виде обыч­ного текста, поскольку это крайне рискованно в отношении безопасности. Кроме того, обычный текст в скомпилированном приложении легко декомпилировать, в результате чего доступ к вашим данным получат лица с сомнительными намерениями.

Предложенный метод реализации безопасности в приложении, которое получает доступ к данным, это использование Windows Authentication (также известного как Integrated Security). Чтобы дополнительно защитить уязвимые данные, связанные с подключением при работе с интегрированной безопасностью, также рекомендуется задать значение False ключевому слову Persist Security Information в строке подклю­чения. Таким образом, верительные данные, которые используются для открытия подключения, сбрасываются, а не хранятся в каком-нибудь месте, откуда их можно извлечь (что делать дальше см. в задании 8).

Как уже говорилось ранее, если вам крайне необходимо использовать строку под­ключения, которая содержит уязвимую информацию, не храните эту строку подклю­чения в компилированном приложении. В качестве альтернативы можно использовать конфигурационный файл (app.config) приложения, который хранит строки подключе­ния в виде XML, а приложение получает информацию о подключении путем запроса этого файла во время выполнения (в отличие от компиляции строки подключения в само приложение). По умолчанию конфигурационный файл приложения хранит информацию в нешифрованном виде, как показано на рис. 1-6.

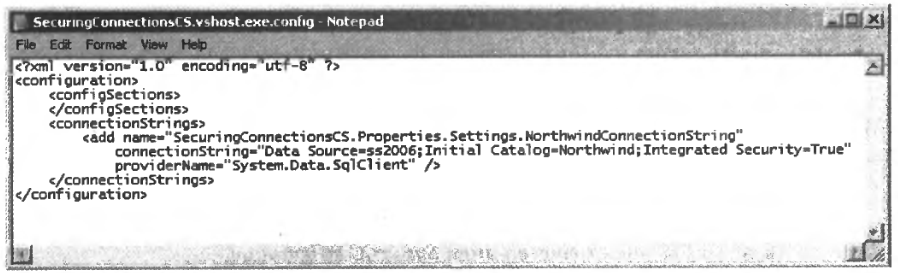


Рис. 1-6

Теперь, когда вы переместили уязвимые данные строки подключения из компилиро­ванного приложения в конфигурационный файл приложения, строка подключения все еще остается нешифрованной и ее может прочитать любой, у кого есть разрешение на открытие конфигурационного файла. То есть вы все еще нуждаетесь в способе, кото­рый бы обеспечил защиту информации о подключении от просмотра посторонними лицами, даже если они каким-то образом получили доступ в конфигурационный файл. Предложенный метод защиты конфигурационных файлов заключается в шифровании разделов, содержащих уязвимую информацию, как показано на рис. 1-7.



Рис. 1-7

Предложенный подход шифрования конфигурационных данных заключается в использовании провайдера с защищенной конфигурацией. В .NET Framework есть два доступных провайдера с защищенной конфигурацией, а также базовый класс, который можно использовать для реализации своего собственного провайдера, если двух доступных окажется недостаточно для вашего приложения.

Шифрование и расшифровывание конфигурационного файла

Это упражнение покажет вам, как с помощью DpapiProtectedConfigurationProvider шифровать и расшифровывать раздел ConnectionStrings файла app.config.

1. Создайте новое приложение Windows и назовите его SecuringConnectionStrings.
2. Добавьте ссылку на пространство имен System.Configuration.
3. Добавьте в форму две кнопки, задав свойствам *Name* и *Text* следующие значения:

|  |  |
| --- | --- |
| Свойство ***Name*** | Свойство ***Text*** |
| *EncryptButton* | Encrypt |
| *DecryptButton* | Decrypt |

1. Создайте источник данных и добавьте строку подключения в конфигурационный файл приложения, запустив мастер Data Source Configuration Wizard.
2. Создайте обработчики события *Click* кнопок.
3. Откройте код и вставьте приведенный ниже код в редактор.

Этот код помещает настройки строки подключения в конфигурационный файл приложения. Настройки строки подключения отмечаются для *шифрования* путем вызова метода *ProtectSection.* Установка значения *True* свойству *ForceSave* обеспе­чивает сохранение конфигурационного файла независимо от того, были внесены какие-либо изменения или нет; а при вызове *Con.figuration.Save* файл сохраняется сразу же после шифрования.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Configuration;

namespace SecuringConnectionsCS

{

public partial class Forml : Form

{

public Form1()

{

InitializeComponent();

}

private void EncryptConnectionString()

{

// Получение конфигурационного файла

System.Configuration.Configuration config = ConfigurationManager.

OpenExeConfiguration(ConfigurationUserLevel.None);

// Создание имени провайдера

string provider = "DataProtectionConfigurationProvider";

//Шифрование строки подключения

ConfigurationSection connstrings = config.ConnectionStrings;

connstrings.Sectionlnformation.ProtectSection(provider);

connstrings.Sectionlnformation.ForceSave = true;

config.Save(ConfigurationSaveMode.Full);

}

private void DecryptConnectionString()

{

// Получение конфигурационного файла

System.Configuration.Configuration config = ConfigurationManager.

OpenExeConfiguration(ConfigurationllserLevel.None);

// Расшифровывание строки подключения

ConfigurationSection connstrings = config.ConnectionStrings;

connstrings.SectionInformation.UnprotectSection();

connstrings.Sectionlnformation.ForceSave = True;

config.Save(ConfigurationSaveMode.Full);

}

private void EncryptButton\_Click(object sender, EventArgs e)

{

EncryptConnectionString();

}

private void DecryptButton\_Click(object sender, EventArgs e)

{

DecryptConnectionString();

}

}

}

1. Запустите приложение и щелкните кнопку Encrypt.
2. Во время работы приложения перейдите к папке проекта и определите местонахож­дение конфигурационного файла (SecuringConnectionStrings.vshost.exe.config).
3. Откройте файл и убедитесь в том, что раздел ConnectionStrings зашифрован.
4. Теперь вернитесь к форме и щелкните кнопку Decrypt.
5. Еще раз откройте файл .config и убедитесь, что строка подключения снова имеет вид обычного текста.

**Контрольные вопросы**

1. Как определить объект Connection, который нужно создать?
2. Какая информация является обязательной для создания объекта Connection?
3. Как нужно открывать подключение к источнику данных?
4. Как нужно закрывать подключение к источнику данных?
5. Каким образом можно узнать, что источник данных возвратил предупреж­дения и информацию?
6. Какая минимальная информация необходима строке подключения, чтобы открыть подключение к базе данных SQL Server 2005 или SQL Server 2008 (выберите все, что применяется)?
7. Допустимый источник данных.
8. Допустимое имя провайдера.
9. Допустимый путь к файлу.
10. Соответствующие верительные данные или настройки Integrated Security.
11. Что происходит при вызове метода Close объекта Connection (выберите все, что применяется)?
12. Соединение разрывается.
13. Соединение возвращается в пул подключений.
14. Происходит событие StateChange.
15. Все непереданные отложенные транзакции откатываются.
16. Какую информацию обычно предоставляет событие InfoMessage?
17. Информацию о текущем состоянии подключения.
18. Грубые ошибки (уровень значимости 17 и выше).
19. Негрубые ошибки SQL Server (уровень значимости 10 и ниже).
20. Сетевые ошибки, которые происходят при попытке открыть подключение.
21. Какие ошибки приведут к закрытию объекта SqlConnection (выберите применяется)?
22. Ошибки с уровнем значимости от 1 до 9.
23. Ошибки с уровнем значимости от 10 до 19.
24. Ошибки с уровнем значимости от 20 до 29.
25. Ошибки с уровнем значимости от 30 и выше.
26. Какое свойство содержит фактическое сообщение об ошибке, которое возвращает SQL Server (выберите все, что применяется)?
27. SqlException.Source.
28. SqlException.Message.
29. SqlError.Class.
30. SqlError.Message.
31. С помощью какого объекта возвращают список видимых SQL Server?
32. VisibleSqlServers.
33. GetDataSources.
34. SqlDataSoureeEnumerator.
35. ServerName.
36. Какая из приведенной ниже информации доступна благодаря объекту SqlServerEnumerator (выберите все, что применяется)?
37. Имя SQL Server.
38. Число баз данных, которые на данный момент находятся на сервере.
39. Версия сервера.
40. Имена экземпляров серверов, на которых не запущены экземпляры по умолчанию.
41. Какая пара ключ/значение строки подключения используется для Windows Authentication в SQL Server 2000 и SQL Server 2005 (выберите все, что применя­ется)?
42. Integrated security - Yes.
43. Integrated Security -SSPI.
44. Integrated Security - True.
45. Trusted\_Connection - Yes.
46. Где вы должны хранить уязвимую информацию в случае необходимости исполь­зования имени пользователя и пароля, ?
47. В приложении в компилированном виде.
48. В зашифрованном конфигурационном файле приложения.
49. В файле ресурса, развернутом вместе с приложением.
50. В системном реестре.
51. Какой метод рекомендуется использовать для защиты уязвимой информации строки подключения?
52. Шифрование данных в конфигурационном файле приложения.
53. Использование программы для шифрования кода.
54. Использование интегрированной безопасности (Windows Authentication).
55. Запрос у пользователя верительных данных во время выполнения.