# Работа с DataSet.

**Объекты DataSet**

Объекты – это наборы данных, которые применяются для временного хранения дан­ных, используемых в вашем приложении. По сути, есть два различных вида объектов *DataSet*: типизированный и нетипизированный.

Нетипизированные объекты *Data­Set* – это стандартные обобщенные экземпляры класса *DataSet,* где вы строите опре­деление (схему) *DataSet,* создавая объекты *DataTable* (нетипизированные) и добавляя их к коллекции *Tables* в *DataSet.* Вы можете обратиться к нетипизированным объектам *DataTable* и *DataColumn* через их индексы в коллекции.

Типизированные объекты *DataSet* наследуют свою схему из файла .xsd и содержат явно типизированные кол­лекции (например, определенного объекта *CustomersTable*).

В Visual Studio существует три различных способа создания объектов *DataSet:*

* объявление нового объекта *DataSet* программно в редакторе кода, что приводит к пустому *DataSet,* требующему создания объектов *DataTable* и необязательного *DataRelation* для добавления к *DataSet;*
* использование инструментов времени разработки, таких как Dataset Designer и Data Source Configuration Wizard, которые помогают в создании типизированных объектов *DataSet,* пошагово продвигая вас через процесс выбора или создания подключения к данным, а затем позволяя выбрать объекты базы данных, доступ­ные в этом подключении, для создания типизированного *DataSet* и генерирования большей части, если не всего необходимого для вас, кода;
* перемещение объекта *DataSet* из Toolbox в форму и использование редакторов коллекций таблиц или столбцов для создания схемы вашего *DataSet.*

На этом занятии описывается, как создавать объекты *DataSet,* используя каждый из перечисленных способов.

**Создание объектов DataSet программно**

Следующий пример кода создает новый *DataSet* с именем *NorthwindDataSet.*

DataSet NorthwindDataSet = new DataSet("NorthwindDataSet");

**Создание объектов DataTable и добавление их к DataSet**

После объявления нового *DataSet* нужно добавить к нему объекты *DataTable.* Следующий пример кода показы­вает, как добавить *CustomersTable* и *OrdersDataTable* к *NorthwindDataSet*

// Создание объектов DataTable

DataTable CustomersTable = new DataTable();

DataTable OrdersDataTable = new DataTable();

// Добавление объектов DataTable к коллекции Tables набора данных

NorthwindDataSet.Tables.Add(CustomersTable);

NorthwindDataSet.Tables.Add(OrdersDataTable);

**Добавление связей между таблицами** *DataSet*

После добавления таблиц к *DataSet* можно создать объекты *DataRelation* для представления связей между объектами *DataTable* (аналогично связям в базе данных).

Следующий пример кода создает в *NorthwindDataSet* связь и предполагает, что оба объекта *DataTable* с именами *Customers* и *Orders* имеют столбец *CustomerlD,* который используется для связывания вместе данных в таблицах:

// Создание новой связи

DataRelation CustomersOrders = new DataRelation("CustomersOrders",

CustomersTable.Columns["CustomerlD"], OrdersTable.Columns["CustomerlD"]);

// Добавление связи к DataSet

NorthwindDataSet.Relations.Add(CustomersOrders);

**Перемещение между связанными таблицами**

Чтобы обращаться к связанным записям объектов *DataTable,* вы должны сначала выбрать *DataRow* в родительской или дочерней таблицах, а затем вызвать метод *GetParentRow* или *GetChildRows* объекта *DataRow.* Вызов метода *GetParentRow* возвра­щает один *DataRow,* представляющий родительскую запись, тогда как вызов метода *GetChildRows* возвращает массив объектов *DataRow,* представляющий все строки, связанные с выбранным родительским элементом.

**Возвращение родительской строки для выбранной дочерней записи** Следующий пример возвращает *Customer* выбранного *Order.*

DataRow Customer = SelectedOrdersRow.GetParentRow("FK\_Orders\_Customers");

**Возвращение связанных дочерних строк для выбранной родительской строки** Сле­дующий пример возвращает *Orders* выбранного *Customer.*

DataRow Orders[] = SelectedCustomersRow.GetChildRows("FK\_Orders\_Customers");

**Объединение содержимого** *DataSet*

Можно взять содержимое одного *DataSet* (исходного набора данных) и объеди­нить его с содержимым другого *DataSet* (целевого набора данных), используя метод *DataSet.Merge.*

При объединении наборов данных существующие данные объединяются в зависи­мости от того, имеется ли подобная запись в *DataSet*, в который они объединяются. Например, если вы объедините два набора данных, оба из которых содержат запись с одним и тем же первичным ключом, значения в целевом *DataSet* будут переписа­ны новыми значениями из исходного *DataSet.* Вы можете управлять этим режимом работы и ограничивать изменения, производимые в целевом *DataSet*, устанавливая в истинное или ложное значение флажок *PreserveChanges* метода *Merge.* В дополне­ние к объединению существующих данных при объединении двух объектов *DataSet*, имеющих таблицы с отличающимися схемами, вы можете передать методу *Merge* необязательный параметр *MissingSchemaAction,* управляющий режимом объединения, когда исходный *DataSet* имеет объекты, которых в настоящее время нет в целевом *DataSet.* Далее указаны допустимые значения параметра *MissingSchemaAction.*

* **Add (по умолчанию)** Все элементы схемы исходного *DataSet* добавляются к це­левому *DataSet* и заполняются.
* **AddWithKey** Все элементы схемы и свойства первичного ключа добавляются к целевому *DataSet.*
* **Error** Когда схемы в исходном и целевом наборах данных не соответствуют, будет вызвано исключение.
* **Ignore** Все противоречия схемы между исходными и целевыми наборами данных игнорируются.

В следующем примере кода содержимое *OldSalesDataSet* объединяется с содер­жимым *SalesHistoryDataSet,* параметр *PreserveChanges* установлен в *True,* любые различия схем будут игнорироваться:

SalesHistoryDataSet.Merge(OldSalesDataSet, true, MissingSchemaAction.Ignore);

## Создание DataSet в Dataset Designer

*Dataset Designer* – это инструмент времени разработки, который помогает в создании типизированных объектов *DataSet,* позволяя вам перетаскивать таблицы базы данных из Server Explorer на поверхность конструктора. Когда вы опускаете таблицы на эту поверхность, они добавляются к *DataSet* в виде типизированных объектов, что делает программирование столь же простым, как и ввод имен таблиц и столбцов, к которым вы хотите обратиться.

1. Создайте приложение Windows и назовите его *DatasetDesignerExample.*
2. В меню Проект выберите Добавить компонент.
3. Выберите шаблон Набор данных (или *DataSet*) и назовите его *DataSetAdventureWorksLT* (рис.1)*.*
4. Перейдите к таблице *Customer* в Server Explorer и перетащите ее на поверхность конструктора (Перед этим необходимо подключиться к вашей базе данных, как в предыдущих лабораторных).
5. Перейдите к таблице *SalesOrderHeader* в Server Explorer и перетащите ее на поверхность конструктора.

После помещения таблиц *Customer* и *SalesOrderHeader* в Dataset Designer вид конс­труктора должен быть похож на рис. 2.

1. Соберите проект перед переходом к следующему шагу.
2. Перетащите *ListBox* в *Forml* и назовите его *CustomersListBox.*
3. Перетащите кнопку в *Forml* и установите такие свойства:

* *Name = GetCustomersButton*
* *Text* = *Загрузить Customer*

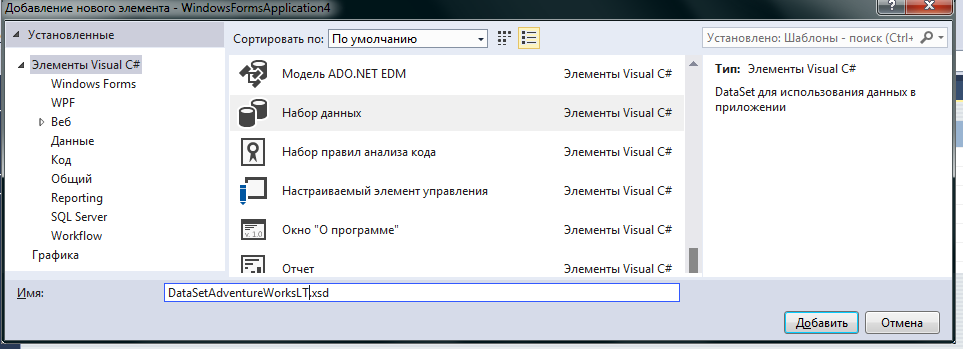


Рис. 1

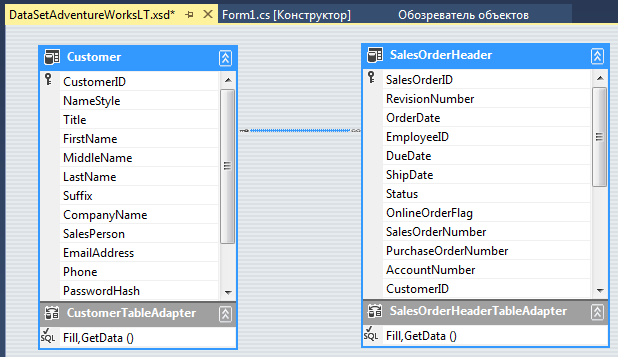


Рис. 2

1. Дважды щелкните кнопку *Загрузить Customers* и добавьте к обработчику события *Get­CustomersButton\_Click* следующий код:

// Создание экземпляра типизированного набора данных AdventureWorksLT

DataSetAdventureWorksLT MyDataSet = new DataSetAdventureWorksLT();

// Создание экземпляра CustomerTableAdapter

DataSetAdventureWorksLTTableAdapters.CustomerTableAdapter CustomerTA = new DataSetAdventureWorksLTTableAdapters.CustomerTableAdapter();

// Вызов метода по умолчанию Fill для загрузки всех клиентов

// в Customers DataTable

CustomerTA.Fill(MyDataSet.Customer);

// Цикл по строкам таблицы Customers и добавление значения

// столбца CompanyName к ListBox

foreach (DataSetAdventureWorksLT.CustomerRow CustRow in MyDataSet.Customer.Rows) CustomersListBox.Items.Add(CustRow.CompanyName);

1. Выполните приложение и щелкните кнопку Загрузить Customers. Убедитесь, что *Compa­nyName* каждого клиента отображается в *CustomersListBox*, как на рис. 3.

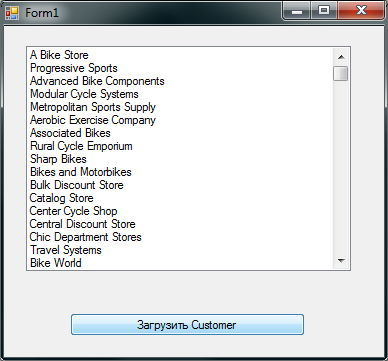


Рис. 3.

## Создание типизированного DataSet в Dataset Source Configuration Wizard

В этом упражнении показано, как создать строго типизированный *DataSet*, используя Data Source Configuration Wizard.

1. Создайте приложение Windows и назовите его *DataSourceWizardExample.*
2. Запустите Data Source Configuration Wizard, выбрав Добавить новый источник данных в меню Проект.
3. На странице Выбор типа источника данных оставьте выбранный по умолчанию База данных, как показано на рис. 3. Затем щелкните Next.
4. Список отображает доступные в Server Explorer подключения к данным. Выберите на странице Choose Your Data Connection, показанной на рис. 4, подключение к примеру базы данных AdventureWorksLT или создайте новое подключение к данным. Затем щелкните Next. Если появится сообщение, щелкните Yes для добавления базы данных к вашему проекту.
5. После выбора желательного подключения к данным отображается установленный флажок для его сохранения в файле настройки приложения. Оставьте флажок по умолчанию, как показано на рис. 5. Затем щелкните Next.

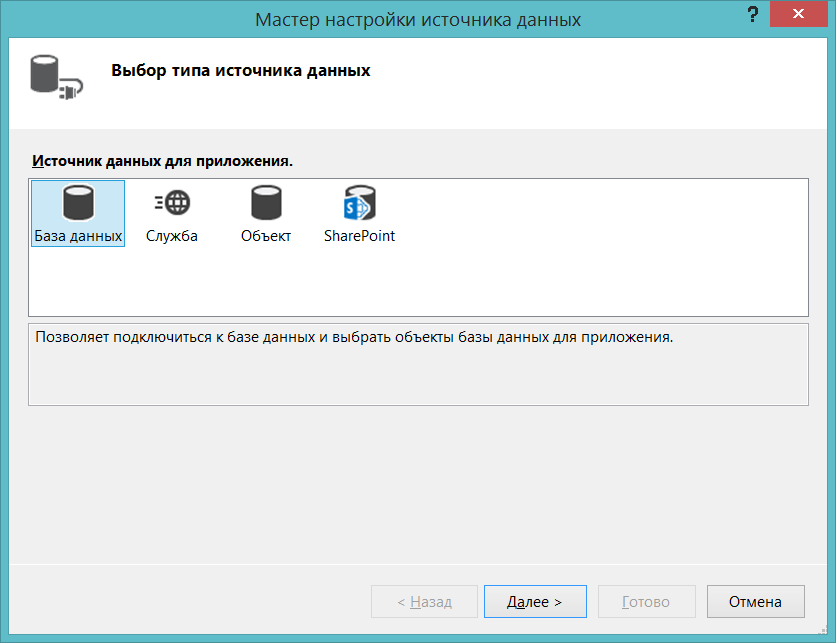


Рис. 3.

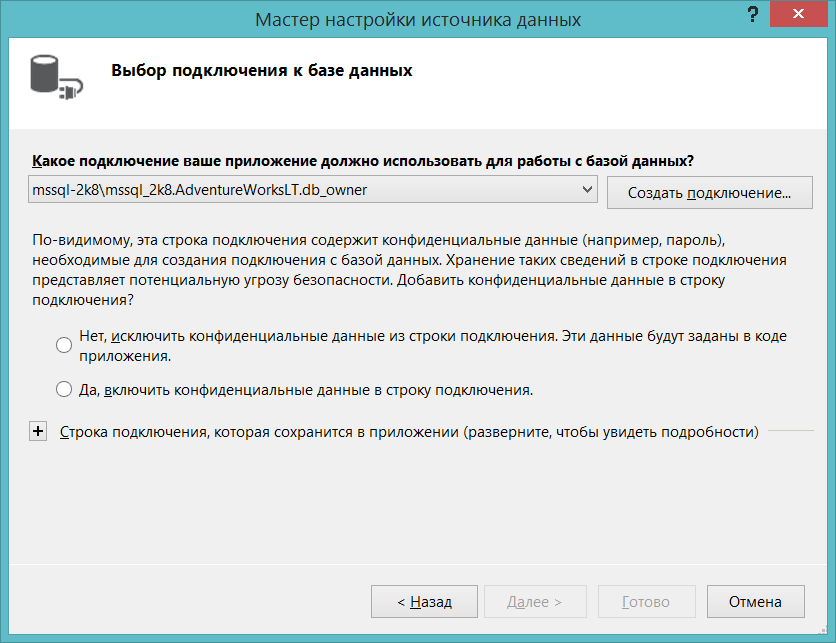


Рис. 4.

1. На странице Choose Your Database Objects разверните узел *Tables* и выберите таб­лицу *Customer,* как показано на рис. 6.
2. Щелкните Finish для завершения работы мастера и добавления типизированного *DataSet* к вашему проекту. Теперь источник данных доступен в окне Data Sources, которое дает возможность конструирования связанных с данными форм.

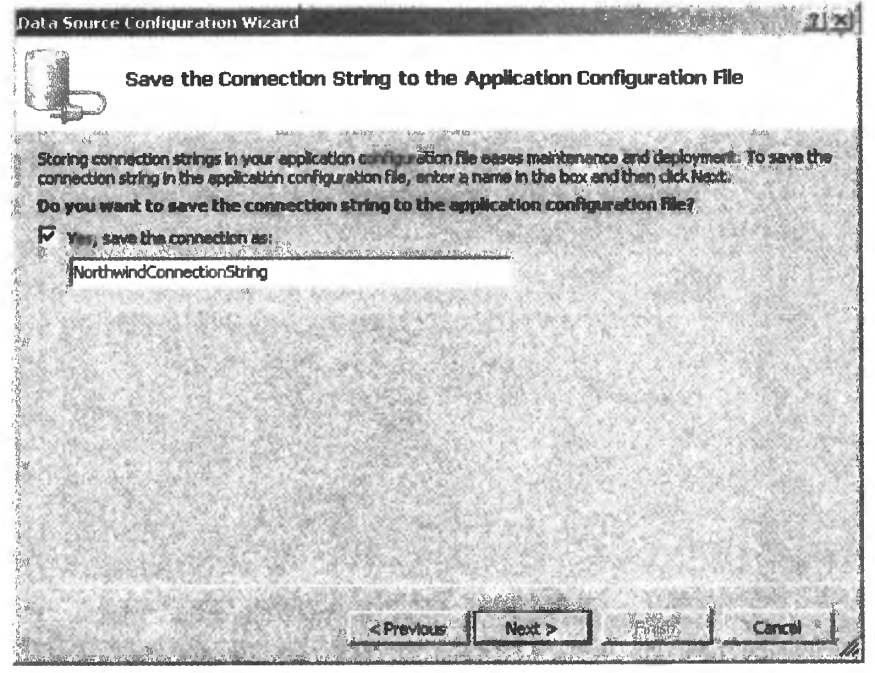


Рис. 5.

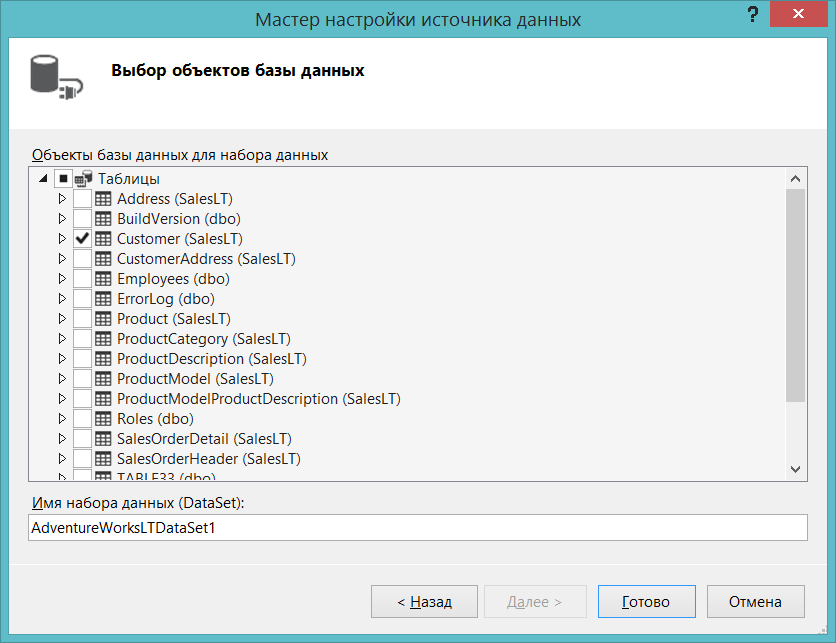


Рис. 6.

1. С выделенной *Form1* в режиме конструктора выберите меню Вид->Другие окна->Источники данных.
2. Перетащите узел *Customer* из окна Источники данных в *Forml* (рис. 7 -7).

В результате перемещения таблицы *Customer* из окна Источники данных Visual Studio добавляет к форме код для заполнения *Customer* (объект *DataTable).*

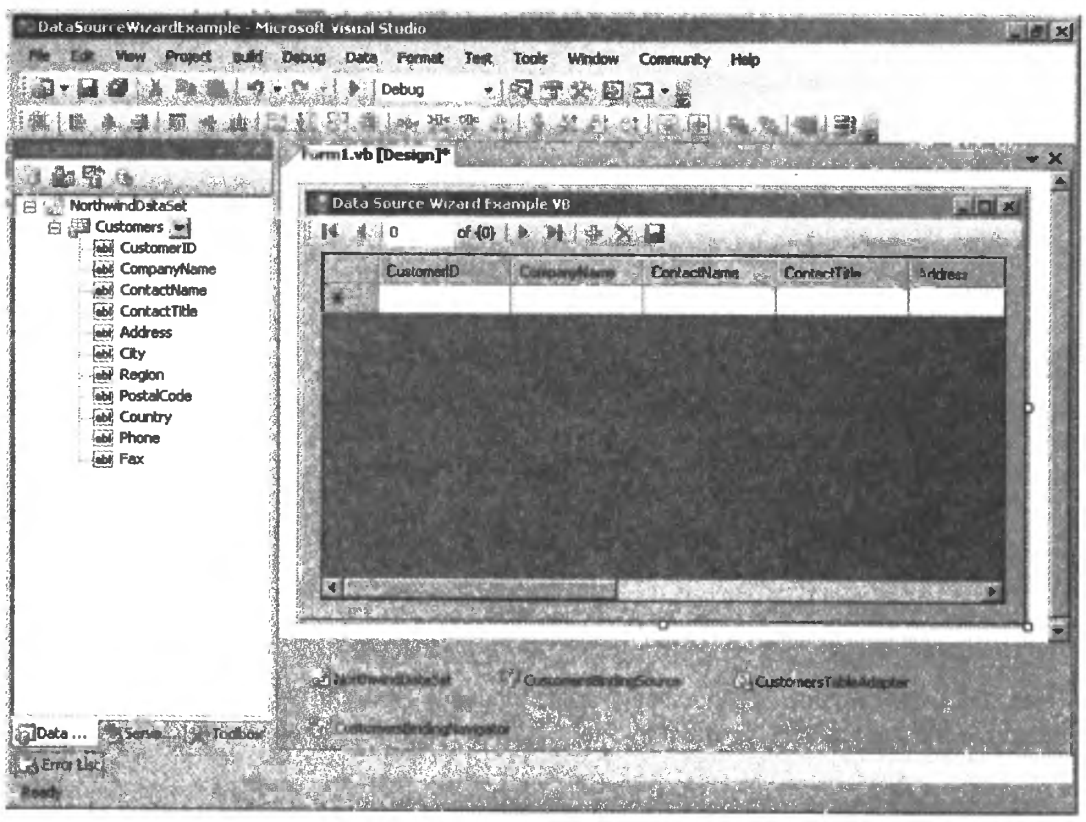


Рис. 7.

1. Выполните приложение и проверьте, появляются ли данные таблицы Customers в форме.

**Объекты DataTable**

На этом занятии описывается, как создавать объекты *DataTable,* являющиеся объек­тами ADO NET в пространстве имен *System.Data,* которые обеспечивают хранение в памяти данных вашего приложения подобно таблице базы данных. *DataTable* могут быть добавлены к *DataSet* и связаны с другими *DataTable* с помощью объекта *DataRe­lation,* или же вы можете использовать *DataTable* как автономные объекты независимо от *DataSet.*

После создания объекта *DataTable* необходимо определить схему таблицы, созда­вая и добавляя столбцы и ограничения подобно таблицам базы данных. Это занятие сфокусировано на создании и настройке *DataTable,* подготавливающих его для полу­чения данных. Процесс заполнения *DataTable* данными, как и управление данными в таблице, будет освещен в этой главе позже.

**Создание объектов** *DataTable*

Создавайте объекты *DataTable,* объявляя экземпляр объекта *DataTable.* Вы можете произвольно указать имя *DataTable,* как и *Namespace,* которое будет использоваться при отображении данных XML. Следующий пример кода создает новый *DataTable* с именем *SalesDataTable:*

Private DataTable SalesDataTable = new DataTableC'SalesData");

**Добавление** *DataTable* **к** *DataSet*

Объекты *DataTable* могут использоваться в качестве автономных объектов, но при отображении в вашем приложении реляционных данных намного полезнее работать с объектами *DataTable* как с частью объектов *DataSet.* В этом случае вы можете свя­зать *DataTable* с объектами *DataRelation* и обращаться к связанным записям так же, как при работе с базой данных. Следующий пример кода создает новый *DataTable* и добавляет его к *DataSet:*

// Создание нового DataSet

DataSet NorthwindDataSet = new DataSet("NorthwindData");

// Создание нового DataTable

DataTable CustomersTable = new DataTable("Customers");

// Добавление DataTable к коллекции Tables в DataSet

NorthwindDataSet.Tables.Add(CostomersTable);

**Создание вычисляемых столбцов в объектах DataTable**

Когда вы хотите хранить в таблице результаты вычислений между существующими столбцами в виде дополнительного столбца, к *DataTable* могут быть добавлены вы­числяемые столбцы. Другими словами, вы можете выполнить вычисления на сущес­твующих столбцах и хранить результаты в дополнительном (вычисляемом) столбце. Рассмотрите, например, таблицу *Order Details* в базе данных Northwind. Таблица содержит столбцы *UnitPrice* и *Quantity*, но не имеет столбца *TotalPrice* (называемого также *ExtendedPrice).* Если вы хотите добавить столбец, отображающий (или храня­щий) общую цену для строки в таблице *Order Details,* сделать это можно добавлением вычисляемого столбца. Следующий пример кода показывает, как создавать вычисля­емый столбец и добавлять его к таблице *NorthwindDataSet. Order Details.*

// Создание нового DataColumn и указание его имени и типа данных

DataColumn TotalPriceColumn = new DataColumn("TotalPrice”,System.Type GetType("System Double"));

// Присвоение свойству Expression этого столбца желаемого выражения,

// в данном случае UnitPrice х Quantity, являющегося общей ценой

// для этой записи

TotalPriceColumn.Expression = ("UnitPrice \* Quantity");

// Добавление столбца к DataTable

northwindDataSet.Order\_Details Columns.Add(Tota'LPriceColiimn);

**Создание автоинкрементных столбцов в объектах DataTable**

Значения автоинкрементному столбцу таблицы назначает сам *DataTable.* Это типично для таблицы, содержащей данные типа сведений о заказе, где желательно иметь уни­кальный *OrderlD,* но действительное значение столбца *OrderlD* не является важным, кроме того, что оно должно быть уникальным. Объекты *DataColumn* предоставляют особые свойства для создания автоинкрементных столбцов, в которых вы можете указать как начальное значение, называемое *AutoIncrementSeed,* так и приращение, на­зываемое *AutoIncrementStep,* — то есть число, добавляемое к последнему значению.

Следующий пример кода показывает, как создать автоинкрементный столбец со значением *SalesOrderlD,* начинающимся со 100 и добавляющим 5 к предыдущему значению для каждой новой добавленной строки. Другими словами, первая созданная запись будет иметь в столбце SalesOrderlD значение 100, следующая строка — значе­ние 105, последующая — 110 и так далее:

// Создание столбца SalesOrderlD и указание

// его типа данных как integer

SalesTable.Columns.Add("SalesOrderlD", Type.GetType("System.Int32"));

// Присвоение свойству Autoincrement значения True делает столбец

// автоинкрементным!

SalesTable.Columns["SalesOrderID"].Autoincrement = true;

// Указание начального значения свойства AutoIncrementSeed SalesTable.Columns["SalesOrderlD”].AutoIncrementSeed = 100;

// Значение AutoIncrementStep определяет величину, добавляемую

// к значению предыдущей строки

SalesTable.Columns["SalesOrderlD"].AutoIncrementStep = 5;

**Добавление ограничений к DataTable**

Объекты *DataTable* могут создаваться с ограничениями подобно таблицам в базах данных. Например, вы можете определить столбцы в *DataTable* в качестве внешних ключей или содержащих уникальные значения, если записи с повторяющимися зна­чениями в столбце являются для таблицы неуместными.

**Создание ограничения внешнего ключа**

Для создания ограничения внешнего ключа создайте экземпляр класса *ForeignKey* и назначьте ограничению желаемый столбец или столбцы родительской и дочерних

таблиц. Например, следующий фрагмент кода создает новое ограничение внешнего ключа между столбцами *OrderlD* таблиц *Orders* и *Order Details* из Northwind:

ForeignKeyConstraint ForeignKey = new

Fo reignKeyConst raint("FK\_0 rde rs\_0 rde rDetaiIs”,

NorthwindDataSet.Orders.Columns["OrderlD"],

NorthwindDataSet.Order\_Details.Columns["OrderlD”]),

NorthwindDataSet.Orders.Constraints.Add(ForeignKey);

**Создание ограничения уникальности**

Для создания ограничения уникальности создайте экземпляр класса *UniqueConstraint* и укажите в конструкторе столбец, содержащий уникальные значения. Например, следующий фрагмент кода помещает ограничение уникальности в столбец *OrderlD* таблицы *Orders* из Northwind:

UniqueConstraint unique = new

UniqueConst raint (NorthwindDataSet:. Orde rs. 0 rde r IDColumn); northwindDataSet.Orde rs.Const raints.Add(unique);

## Создание объектов DataTable

В следующем примере описано создание *DataTable,* а также создание новой строки данных и добавление ее в таблицу. Для наглядности *DataTable* отображается в *Data- GridView.*

1. Создайте приложение Windows и назовите его *CreatingDataTables.*
2. Добавьте *DataGridView* к *Forml* и назовите его *TableGrid.*
3. Добавьте к форме кнопку и установите следующие свойства:

* *Name* - *CreateTableButton*
* *Text* - *Create Table*

1. Добавьте к форме вторую кнопку и установите следующие свойства:

* *Name* = *AddRowButton*
* *Text = Add Row*

1. Дважды щелкните каждую кнопку для создания обработчиков события щелчка кнопки.
2. Добавьте к форме нижеприведенный код (глобальный в форме, то есть вне любых методов):

// Создание нового экземпляра таблицы (глобально в форме)

private DataTable CustomersTable = new DataTable("Customers");

1. Добавьте следующий код к обработчику события *CreateTableButton jClick*:

// Присвоение DataGridView отображаемой таблицы

TableGrid.DataSource = CustomersTable;

// Определение схемы таблицы добавлением объектов DataColumn

// к коллекции Columns этой таблицы

CustomersTable.Columns.Add("CustomerlD", Type.GetType("System.String")); CustomersTable.Columns.Add("CompanyName", Type.GetType("System.String")); CustomersTable.Columns.Add("ContactName", Type.GetType("System.String")); CustomersTable.Columns.Add("ContactTitle", Type.GetType("System.String")); CustomersTable.Columns.Add("Address", Type.GetType("System.String")); CustomersTable.Columns.Add("City", Type.GetType("System.String")); CustomersTable.Columns.Add("Region", Type.GetType("System.String")); CustomersTable.Columns.Add("PostalCode", Type.GetType("System.String")); CustomersTable.Columns.Add("Country", Type.GetType("System.String")); CustomersTable.Columns.Add("Phone", Type.GetType("System.String")); CustomersTable.Columns.Add("Fax", Type.GetType("System.String"));

// Указание столбца CustomerlD как первичного ключа

DataColumn[] KeyColumns = new DataColumn[1];

KeyColumns[0] = CustomersTable.Columns["CustomerlD"];

CustomersTable.PrimaryKey = KeyColumns;

// Установка запрета значений Null для столбцов CustomerlD и CompanyName CustomersTable.Columns["CustomerlD"].AllowDBNull = false;

CustomersTable.Columns["CompanyName"].AllowDBNull = false;

1. Добавьте следующий код к обработчику события *AddRowButton\_Click\*

// Создание записи (DataRow) для добавления в таблицу

DataRow CustRow = CustomersTable.NewRow();

Object[] CustRecord = {"ALFKI", "Alfreds Futlerkiste", "Maria Anders",

"Sales Representative", "Obere Str. 57", "Berlin", null, "12209", "Germany", "030-0074321", "030-0076545"};

CustRow.ItemArray = CustRecord;

// Добавление записи в таблицу

CustomersTable.Rows.Add(CustRow);

1. Выполните приложение и щелкните кнопку Create Table.

Сетка отображает пустую таблицу данных со столбцами, как показано на рис. 7-9.

1. Щелкните кнопку Add Row, и в таблицу добавится новая запись, как показано на рис. 7-10.

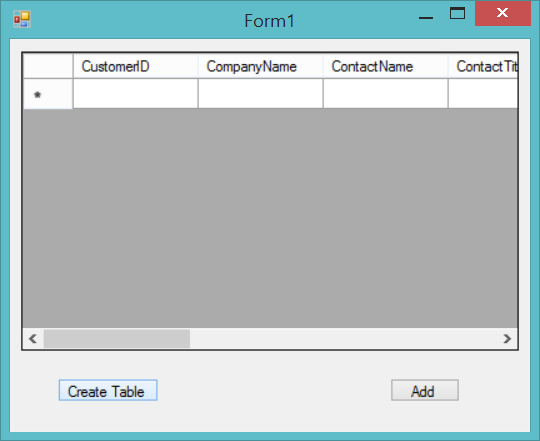


Рис. 9.

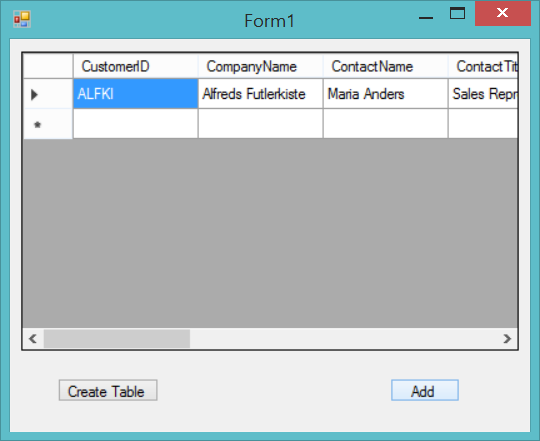


Рис. 10.

1. Щелкните кнопку Add Row снова — и вы увидите произошедшее исключение.

**Контрольные вопросы**

* + - 1. Что такое *DataSet* (выберите все, что применяется)?
  1. Указатель на удаленную базу данных.
  2. Коллекция объектов DataTable и DataRelation.
  3. Кэш данных в памяти.
  4. Коллекция записей базы данных.
     + 1. Какие три основных объекта используются при работе с *DataSet?*

1. *DataTable, DataRelation* и *DataAdapter.*
2. *DataTable, DataColumn* и *DataRelation.*
3. *DataTable, DataRelation* и *Constraint.*
4. *DataTable, DataColumn* и тип.
   * + 1. Как программно обратиться к связанным записям в *DataSet?*
5. Вызывая методы *GetParentRow* и *GetChildRows* объекта *DataSet.*
6. Вызывая методы *GetParentRow* и *GetChildRows* объекта *DataTable.*
7. Вызывая методы *GetParentRow* и *GetChildRows* объекта *DataRow.*
8. Обращаясь к *ParentColumns* и *ChildColumns* объекта *DataRelation.*
   * + 1. Какие действия нужны для создания *DataTable* программно?
9. Выполнить Data Source Configuration Wizard.
10. Создать новый экземпляр *DataTable* и добавить объекты *DataColumn* к коллекции *DataTable.Columns.*
11. Добавить новый объект *DataSet* к вашему проекту. Г. Создать новый экземпляр объекта *DataSet.*
    * + 1. Свойства какого объекта вы настроите, чтобы создать первичный ключ для *Data-Table!*
12. DataSet*.* Б. *DataTable.*
13. DataColumnГ. *DataRelation.*
    * + 1. Как указать, что *DataColumn* в *DataTable* является частью внешнего ключа?
14. Присвоить свойству *DataColumn.ForeignKey* значение *True.*
15. Присвоить свойству *DataTable.ForeignKey* желаемый *DataColumn.*
16. Создать новый экземпляр класса *ForeignKey* и добавить его к коллекции *Data-Table. Constraints.*
17. Определить внешний ключ в объекте DataRelation.