# Создание связанных с данными элементов управления.

Связывание данных позволяет доставить их в ваше прило­жение, обычно в набор данных, а затем изменить эти данные, редактируя значения в связанных с данными элементах управления и отправляя изменения обратно в базу данных. Инфраструктура связывания данных обрабатывает большинство закулисных задач, таких как хранение элемента управления синхронизированным с *DataTable.*

## Создание связанной с данными формы в мастере источников данных

Создадим приложение Windows и покажем создание источника данных и связывание элементов управления с данными, перетаскивая элементы из окна Data Sources.

1. Создайте приложение Windows и назовите его *DataSourcesWizardExample.*
2. Запустите Data Source Configuration Wizard (Менеджер источников данных), выбрав Add New Data Source(Добавить новы источник данных в меню Data (Источники данных).
3. На странице Choose a Data Source Type (Выбрать источник данных) оставьте выбранный по умолчанию Database и щелкните Next.
4. На странице Choose Your Data Connection (Выбор подключения к базе данных) выберите подключение к вашей базе данных или создайте, если нужно, новое подключение.
5. Щелкайте Next, сохраняя значения по умолчанию, пока не дойдете до страницы Choose Your Database Objects (Выбор объектов базы данных), и в узле *Tables (Таблицы)* выберите следующее:

* *Customer*
* *SalesOrderHeader*
* *SalesOrderDetail*

1. Щелкните Finish для добавления набора данных к вашему проекту.
2. В меню View (Вид) выберите Show Data Sources (Источники данных), чтобы отобразить окно Data Sources (Источники данных). Окно Data Sources (Источники данных) должно выглядеть, как на рис. 1.

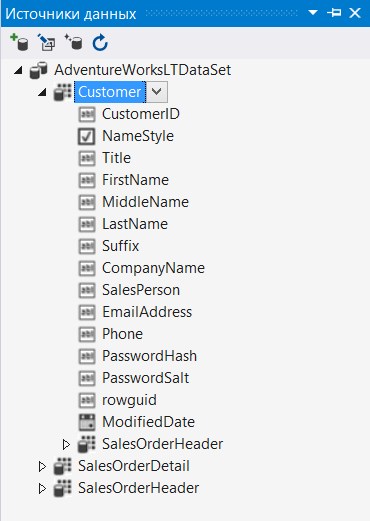


Рис. 1. Окно Data Sources

1. Перетащите узел *Customer* из окна Data Sources в *Form1.*

К форме добавятся *DataGridView* и *BindingNavigator,* и в области компонентов появится несколько относящихся к данным объектов, как показано на рис. 2.

1. Выполните приложение.

В этот момент вы имеете рабочее приложение с *DataGridView,* связанным с данны­ми таблицы *Customer.* Если в интегрированной среде разработки вы переключитесь в режим кода, то увидите, что был добавлен код к событию загрузки формы для заполнения таблицы *Customer* данными, а в *BindingNavigator* — к сохраняющей данные кнопке для отправки обновлений обратно в базу данных.

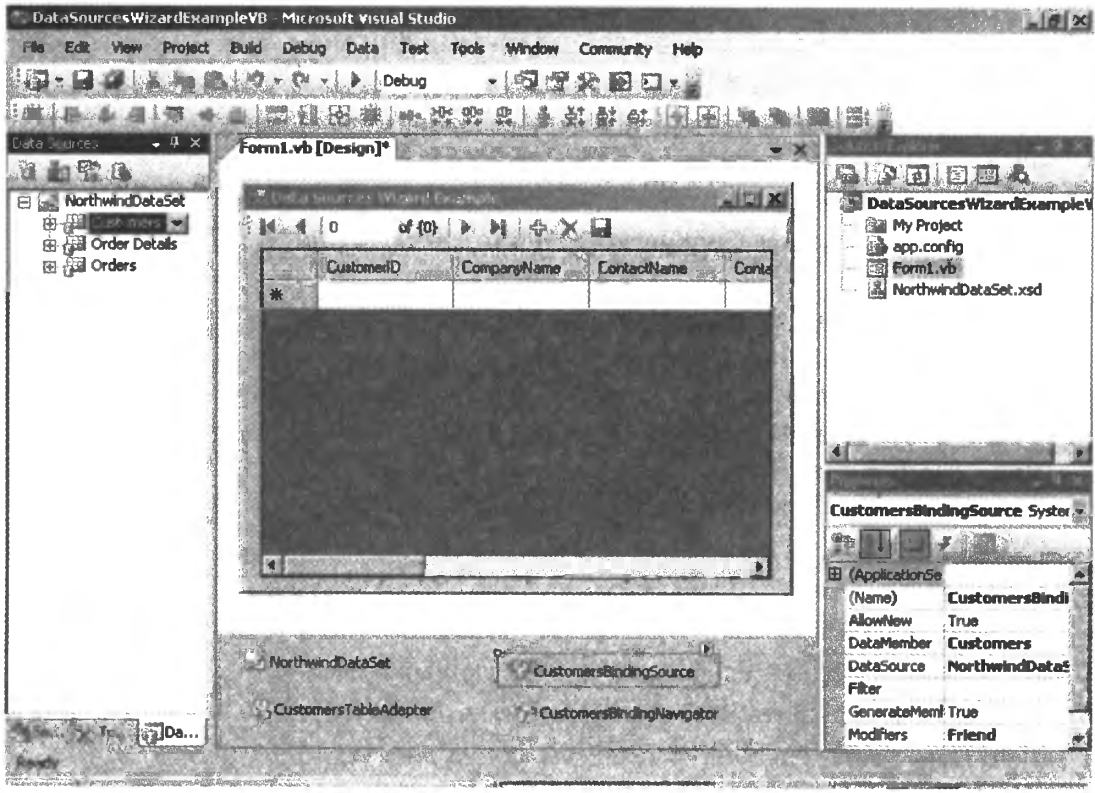


Рис. 2. Form1 и дополнительные компоненты после перемещения узла Customer из окна Data Sources

Выполняющееся приложение должно отобразить данные *Customer,* как показано на рис. 3.

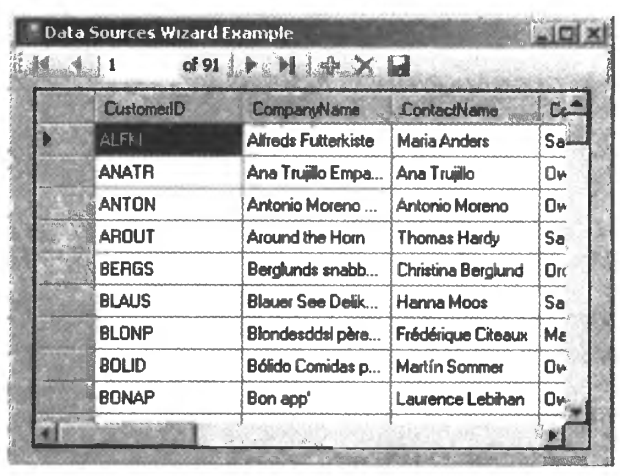


Рис. 3. Формы отображающая данные из таблицы Customers

1. Остановите приложение и откройте форму в режиме Design
2. Разверните узел *Customer* в окне Data Sources
3. Перетащите в форму узел *SalesOrderHeader* (вложенный в узел *Customer).*

Обратите внимание на *SalesOrderHeaderBindingSource* и *SalesOrderHeaderTableAdapter,* добавленные в область компонентов.

1. Выполните приложение.

Щелкните строку в таблице *Customer.* Обратите внимание, что *SalesOrderHeader* *DataGridView* отображает все заказы выбранного клиента, как показано на рис. 4.

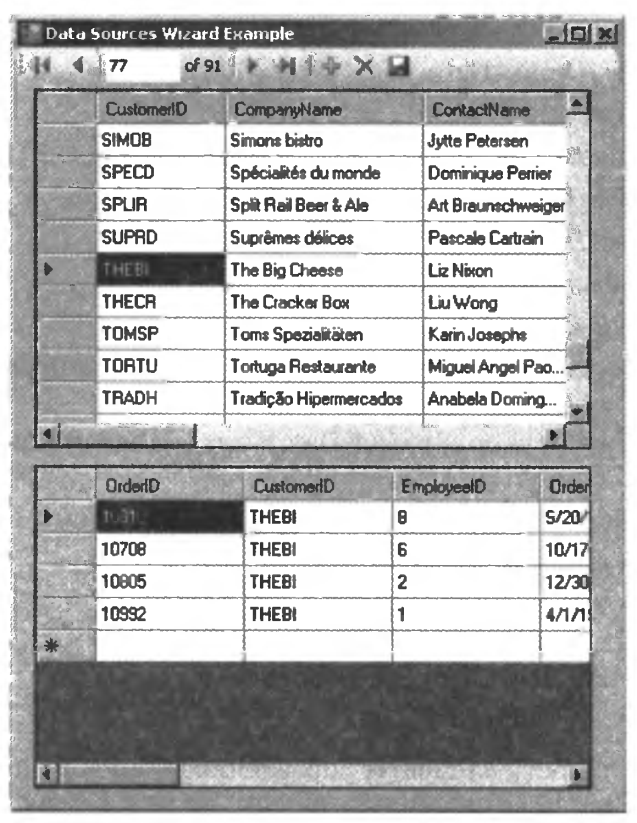


Рис. 4. Форма, отображающая данные Sales*OrderDetail* для выбранного клиента.

**Создание связанных с данными элементов управления**

Вы можете связать простые элементы управления, например *TextBox, Label* или даже *MonthCalendar, с* одним элементом данных или связать более сложные элементы управления, напри­мер *DataGridView* и *ComboBox,* с несколькими элементами данных.

Связывание элементов управления с данными является просто описанием процесса отображения данных (таких как данные из базы данных) в элементах управления Windows Forms.

Простое связывание данных описывает отображение одного элемента данных в элементе управления, например отображение в *TextBox* значения одного столбца таблицы, типа названия компании.

Сложное связывание данных описывает связывание элемента управления больше чем с одним источником данных. Например, рассмотрим поле со списком, которое отображает названия категорий. Предположим, что в таблице, которую вы отобра­жаете, есть только идентификатор категории, как в таблице *Products* примера базы данных Northwind. В этом случае для отображения значения столбца одного *DataTable* на основании значения внешнего ключа в другом *DataTable* вы можете использовать сложное связывание данных.

Простое связывание данных.

Простое связывание данных связывает один элемент данных с одним свойством элемента управления, например отображение *TextBox* (в его свойстве *Text)* столбца *ProductName* таблицы.

Следующий код показывает, как связать столбец *ProductName* из *DataTable* с *TextBox* под именем *TextBoxI*:

TextBoxI.DataBindings.Add("Text", productsBindingSource, "ProductName");

Сложное связывание данных

Сложное связывание данных связывает больше чем один элемент данных больше чем с одним свойством элемента управления. В качестве примера можно привести отображение элементом управления *DataGridView* всей таблицы или элементом уп­равления *List* нескольких столбцов данных.

Элементы управления, которые допускают сложное связывание данных, как пра­вило, содержат свойства *DataSource* и *DataMember.* Свойство *DataSource* обычно яв­ляется объектом *BindingSource* или *DataSet,* а свойство *DataMember* — фактически отображаемой таблицей или коллекцией.

Следующий код показывает, как, используя компонент *BindingSource,* связать *DataGridView* с таблицей *Customers* из Northwind:

BindingSource customersBindingSource = new BindingSource(northwindDataSet1, "Customers");

DataGridViewl.DataSource = customersBindingSource;

Следующий код показывает, как связать *DataGridView с* таблицей *Customers* из Northwind, используя набор данных:

DataGridViewl.DataSource = northwindDataSetl;

DataGridViewl.DataMember = "Customers";

Перемещение по записям в ***DataSet***

Для перемещения по записям в источнике данных используйте компонент *Binding- Navigator\* Назначьте свойству *BindingNavigatorBindingSource* допустимый компонент *BindingSource,* после чего *BindingNavigator* может использоваться для перемещения назад и вперед по записям в этом источнике данных.

Для перемещения по записям *BindingNavigator* использует методы перемещения, имеющиеся в *BindingSource.* Например, в *BindingSource* доступны методы *MoveNext* и *MovePrevious.*

Определение источника данных с использованием компонента ***BindingSource***

Компонент *BindingSource* содержит сведения, необходимые элементам управления для связывания с *BindingSource* посредством передачи им ссылки на *DataTable* в *DataSet.* Связывая с *BindingSource* вместо *DataSet,* вы можете легко перенаправить ваше при­ложение к другому источнику данных без необходимости перенаправлять к новому источнику данных весь связывающий данные код.

Следующий код показывает, как создать *BindingSource* и назначить ему ссылку на таблицу *Customers* из Northwind:

customersBindingSource = New BindingSource(northwindDataSet1, "Customers");

## Простое связывание данных

В этом упражнении создается приложение Windows и показывается, как выполнить простое связывание с данными элементов управления. В примере связывание данных настраивается в коде (в противоположность перемещению элементов из окна Data Sources).

1. Создайте приложение Windows и назовите его *SimpleDataBindingExample.*
2. В окне Data Sources (Источники данных) щелкните Add New Data Source (Добавить новый источник данных) (создает новый типизирован­ный набор данных, выполняя Data Source Configuration Wizard).
3. Оставьте выбранный по умолчанию Database (на странице Choose a Data Source Туре) и щелкните Next.
4. На странице Choose Your Data Connection выберите подключение к вашей базе данных или создайте, если нужно, новое подключение.
5. Щелкайте Next, сохраняя значения по умолчанию, пока вы не дойдете до страницы Choose Your Database Objects (Выберите объекты базы данных), где выберите *Product* в узле *Tables (Таблицы).*
6. Щелкните Finish для добавления набора данных к вашему проекту.
7. Соберите проект.
8. Перетащите объект *AdventureWorksLTDataSet* из Toolbox (Панель элементов) в форму (вы должны собрать проект, чтобы *AdventureWorksLTDataSet* появился в Toolbox).
9. Перетащите объект *ProductsTableAdapter* из Toolbox в форму.

Теперь, когда мы имеем набор данных (и *TableAdapter* для его заполнения), созда­дим несколько элементов управления для отображения данных набора данных *(AdventureWorksLTDataSet*) и перемещения по этим данным.

1. Добавьте элемент управления *TextBox* к форме и присвойте его свойству *Name* значение *ProductIDTextBox.*
2. Добавьте еще один *TextBox* к форме и присвойте его свойству *Name* значение *Pro- ductNameTextBox.*

Добавим пару кнопок для перемещения по записям.

1. Добавьте элемент управления *Button* и установите следующие свойства:

* *Name* - *PreviousButton*
* *Text* = *Previous*

1. Добавьте еще один элемент управления *Button* и установите следующие свойства: a *Name* = *NextButton*

* *Text* = *Next*

1. Дважды щелкните пустую область в форме и создайте обработчик события *Form1\_ Load.*
2. Добавьте код для создания в форме глобального *BindingSource* и затем добавьте код к обработчику события *FormI Load* для настройки связывания данных, чтобы ваш код формы был похож на следующий:

// Объявление BindingSource для таблицы Products private BindingSource productsBindingSource

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

{

// Заполнение таблицы Products данными ProductsTableAdapterl.Fill(*AdventureWorksLT*DataSet1.Products);

// Создание BindingSource для таблицы Products

productsBindingSource = new BindmgSource(*AdventureWorksLT*DataSet1,"Products");

// Настройка связывания для TextBox

ProductIDTextBox.DataBindings.Add("Text", productsBindingSource, "ProductlD');

ProductNameTextBox.DataBindings.Add("Text", productsBindingSou rce, "ProductName");

}

1. Дважды щелкните кнопку Previous и добавьте следующий код, который перемещает к предыдущей записи в источнике данных *BindingSource*:

productsBindingSource.MovePrevious();

1. Дважды щелкните кнопку Next и добавьте указанный ниже код, который переме­щает к следующей записи в источнике данных *BindingSource*:

productsBindingSource.MoveNext();

1. Выполните приложение.

Щелкните кнопки Next и Previous для перемещения назад и вперед по данным в таблице *Products.* Каждое текстовое поле связано с одним столбцом в таблице данных, и объект *BindingSource* управляет обслуживанием низкого уровня.

## Сложное связывание данных

В этом упражнении создается приложение Windows и показывается, как выполнить сложное связывание с данными элемента управления (*DataGridView*). В примере связывание данных настраивается в коде (в противоположность перемещению эле­ментов из окна Data Sources).

* + - 1. Создайте приложение Windows и назовите его *ComplexDataBindingExample.*
      2. В окне Data Sources щелкните Add New Data Source (создает новый типизирован­ный набор данных, выполняя Data Source Configuration Wizard).

3 Оставьте выбранный по умолчанию Database (на странице Choose a Data Source Туре) и щелкните Next.

1. На странице Choose Your Data Connection выберите подключение к вашей базе данных или создайте, если нужно, новое подключение.
2. Щелкайте Next, сохраняя значения по умолчанию, пока вы не дойдете до страницы Choose Your Database Objects, где выберите *Product* в узле *Tables.*
3. Щелкните Finish для добавления набора данных к вашему проекту.
4. Соберите проект.
5. Перетащите объект *AdventureWorksLTDataSet* из Toolbox в форму (вы должны собрать проект, чтобы *AdventureWorksLTDataSet* появился в Toolbox).
6. Перетащите *ProductTableAdapter* из Toolbox в форму.
7. Перетащите *DataGridView* в форму и присвойте его свойству *Name* значение *Pro­ductGrid.*
8. Перетащите элемент управления *Button* в форму, а затем установите следующие свойства:

*Name* - *BindGridButton*

*Text* = *Bind Grid*

1. Дважды щелкните кнопку и добавьте следующий код обработчика события *Click* для *Button*:

// Заполнение таблицы Products данными из базы данных ProductsTableAdapterl.Fill(*AdventureWorksLT*DataSet1.Products);

// Создание нового компонента BindingSource

BindingSource ProductsBindingSource = new BindingSource(*AdventureWorksLT*DataSet1, "Products');

// Связывание сетки с компонентом BindingSource ProductsGrid.DataSource = ProductsBindingSource;

1. Перетащите *BindingNavigator* в форму.
2. Добавьте следующую строку кода в конец добавленного ранее кода события *Click*

для *Button*:

// Привяжите навигатор к BindingSource

bindingNavigatorl.BindingSource = ProductsBindingSource;

1. Выполните приложение.
2. Щелкните кнопку Bind Grid — и данные таблицы *Products* загрузятся в элемент управления *DataGridView.* Вы можете переместиться назад и вперед, применяя *BindingNavigator,* потому что он настроен для использования того же объекта *Bind­ingSource,* что и сетка. Поскольку объект *BindingSource* уже настроен для исполь­зования таблицы *Products,* необходимо только присвоить свойству *DataGridView. DataSource* объект *ProductsBindingSource,* без установки свойства *DataMember.*

## Работа с DataGridView

Создадим приложение Windows и покажем, как управлять определением элемента управления *DataGridView,* его столбцами и данными.

1. Создайте приложение Windows и назовите его *DataGridViewExample.*
2. Откройте окно Data Sources (в меню Data выберите Show Data Sources).
3. Щелкните Add New Data Source, для того чтобы запустить Data Source Configura­tion Wizard.
4. Оставьте выбранный по умолчанию Database и щелкните Next.
5. Выберите (или создайте) подключение к вашей базе данных и щелк­ните Next.
6. Разверните узел *Tables.* Выберите таблицу *Customer* и затем щелкните Finish.
7. Перетащите узел *Customer* из окна Data Sources в форму.

В этот момент вы уже можете выполнить приложение, при этом форма появится с таблицей *Customer*, загруженной в *DataGridView.*

1. Перетащите две кнопки в форму ниже *DataGridView* и установите следующие свойства:

*Button1*

* *Name* - *AddColumnButton*
* *Text* = *AddColumn*

*Button2*

* *Name* - *DeleteColumnButton*
* *Text = Delete Column*

1. Дважды щелкните кнопку Add Column, чтобы создать обработчик события щелчка кнопки и открыть форму в режиме кода.
2. Включите следующий код в *Forml,* добавляющий в событие *Form1\_Load* дополни­тельный код для создания нового столбца в *DataTable,* а в событие *AddColumnBut- tonjClick —* код для добавления нового столбца в *DataGridView*:

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

{

// Добавление нового столбца к Customers DataTable, использующегося

// для демонстрации добавления и удаления столбцов в DataGridView

// методами ниже

DataColumn Location = new DataColumn("Location");

Location.Expression = "City + + Country";

AdventureWorksLTDataSet.Custome rs.Columns.Add(Location);

}

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

{

DataGridViewTextBoxColumn LocationColumn = new DataGridViewTextBoxColumn();

LocationColumn.Name = "LocationColumn";

LocationColumn.HeaderText = "Location";

LocationColumn.DataPropertyName = "Location";

CustomerDataGridView.Columns.Add(LocationColumn);

}

1. Дважды щелкните кнопку Delete Column, чтобы создать обработчик события *DeleteColumnButton Click.* Добавьте к нему следующий код;

try

{

CustomerDataGridView.Columns.Remove("LocationColumn”);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

1. Перетащите еще одну кнопку в форму и установите следующие свойства:

*Name* = *GetClickedCellButton*

*Text* = *Get Clicked Cell*

1. Перетащите надпись в форму и поместите ее рядом с кнопкой Get Clicked Cell.
2. Дважды щелкните кнопку Get Clicked Cell и добавьте к обработчику события *GetClickedCellButton Click* следующий код:

string CurrentCelllnfo;

CurrentCelllnfo = CustomerDataGridView.CurrentCell.Value.ToStringO + Environment.NewLine;

CurrentCelllnfo += "Column: " +

CustomerDataGridView.CurrentCell.OwningColumn.DataPropertyName +

Envi ronment.NewLine;

CurrentCelllnfo += "Column Index: " +

CustomerDataGridView.CurrentCell.Columnlndex.ToStringO +

Environment.NewLine;

CurrentCelllnfo += "Row Index: " +

CustomerDataGridView. CurrentCell. Rowlndex.ToStringO;

labell.Text = CurrentCelllnfo;

1. Создайте обработчик для события *CustomerDataGridViewjCellValidating* (выбе­рите в форме *CustomerDataGridView*, щелкните кнопку Events в окне Properties и дважды щелкните событие *CellValidating').*
2. Добавьте к обработчику события *CellValidating* следующий код:

if (customerDataGridView.Columns[e.Columnlndex] DataPropertyName == "ContactName")

{

if (e.FormattedValue.ToString() == "")

{

customerDataGridView.Rows[e.Rowlndex].ErrorText =

"ContactName is a required field"; e.Cancel = true;

}

else

customerDataGridView.Rows[e.Rowlndex].ErrorText = "";

}

1. Перетащите еще одну кнопку в форму и установите следующие свойства:

*Name* - *ApplyStyleButton*

*Text* — *Apply Style*

1. Дважды щелкните кнопку Apply Style и добавьте к обработчику события *Apply- StyleButtonjClick* следующий код:

customerDataGridView.AlternatingRowsDefaultCellStyle.BackColor = Color.LightGray;

1. Выполните приложение.
2. Щелкните кнопку Add Column и затем перейдите в конец столбцов для проверки того, что новый столбец *Location* существует.
3. Щелкните кнопку Delete Column и проверьте, удален ли столбец *Location* из *Da­taGridView.*
4. Выберите любую ячейку в сетке и затем щелкните кнопку Get Clicked Cell. Надпись отображает содержимое ячейки, имя столбца ячейки и индекс столбца и строки ячейки.
5. Наконец, щелкните кнопку Apply Style, после чего будет установлен *Altematin- gRowCellStyle* для отображения чередующихся строк со светло-серым фоном.

**Контрольные вопросы**

* + - 1. Как добавить элементы к окну Data Sources (выберите все, что применяется)?
  1. Выбрать Add New Data Source в окне Data Sources. Б. Перетащить *DataSet* из Toolbox в форму.
  2. B. Выбрать Add New Data Source в меню Data. Г. Выполнить Data Source Configuration Wizard.
     + 1. Как создать связанные с данными элементы управления в форме Windows?

1. Перетащить элементы из Server Explorer в Dataset Designer. Б. Перетащить элементы из окна Data Sources в форму.
2. Перетащить элементы из Toolbox в форму.
3. Настроить относящиеся к данным свойства *Form* в окне Properties.
   * + 1. Как связать элементы управления для отображения в форме Windows данных из связанных таблиц?
4. Выбирая основной узел связанной таблицы в окне Data Sources и перетаскивая его в форму.
5. Выбирая дочерний узел в окне Data Sources и перетаскивая его в форму.
6. Перетаскивая объект Relation в Dataset Designer. Г.
7. Связывая дочерний DataTable в DataSet.
   * + 1. Что означает для элемента управления простое связывание данных?
8. Элемент управления отображает только один столбец данных.
9. Элемент управления отображает данные только в виде строк, как в TextBox или ListBox.
10. Элемент управления может отображать лишь простые типы данных.
11. Элемент управления имеет одно свойство, связанное с одним столбцом дан­ных.
    * + 1. Какие нужны две основные части сведений, когда используется сложное связы­вание данных с элементом управления?
12. Имя базы данных и имя таблицы с данными..
13. Имя базы данных и имя набора данных.
14. Объект источника данных, например *BindingSource* или *DataSet,* и отображае­мый список, например *DataTable.*
15. Имя DataSetи имя элемента управления.
    * + 1. Зачем используется компонент *BindingSource* (выберите все, что применяется)?
16. Компонент *BindingSource* предоставляет отвлеченный уровень между связан­ными элементами управления и источником данных, упрощая процесс перена­правления вашего приложения на использование разных источников данных.
17. Компонент *BindingSource* предоставляет методы для перемещения назад и впе­ред в *DataTable.*
18. Компонент *BindingSource* содержит методы, необходимые для отправки обнов­лений назад и вперед между приложением и базой данных.
19. BindingSourceпредоставляет события, с помощью которых вы можете добавить к вашему коду логику проверки правильности данных.
    * + 1. Какой способ является лучшим для определения ячейки, щелкнутой пользователем в *DataGridView?*
20. Использовать индексы столбца и строки выбранной ячейки.
21. Использовать свойство *DataGridView.CurrentCell.*
22. Использовать координаты курсора *х* и *у.*
23. Использовать выбранные в настоящее время столбец и строку в связанном *DataTable* для определения щелкнутой ячейки.
    * + 1. Какой способ проверки ввода в *DataGridView* более предпочтителен?
24. Добавление проверочного кода в обработчик события *CellPainting.*
25. Добавление проверочного кода в обработчик события *DataGridView\_Cell-Click.*
26. Добавление проверочного кода в обработчик события *DataGridViewjCellValida-ting.*
27. Добавление кода в файл частичного класса *DataGridView.* 
    * + 1. Каким способом лучше отобразить булево значение в *DataGridView?*
28. Настроить *DataGridViewTextBoxColumn* и отобразить *True* или *False.*
29. Настроить *DataGridViewCheckBoxColumn* для отображения установленного или снятого флажка.
30. Настроить *DataGridViewButtonColumn* для отображения нажатой или отжатой кнопки.
31. Настроить специализированный столбец для отображения *Да* или *Нет*