ПРИВЯЗКА ДАННЫХ К ЭЛЕМЕНТАМ ИНТЕРФЕЙСА

Привязка данных дает возможность отображать и изменять информацию из источника данных в интерфейсе пользователя. Можно выполнить привязку данных не только из традиционных источников данных, но и практически из любых структур, содержащих данные.

Простая привязка данных в формах Windows

В проектах, построенных на основе Windows-форм, используют ***простую*** (simple data binding) и ***составную*** (complex data binding) привязки данных. ***Простой привязкой данных*** (simple data binding) называется связывание элемента управления с единственным элементом данных.

Для реализации привязки данных удобно использовать класс ***BindingSource***, который является своеобразным контроллером между элементом управления и данными, обеспечивает навигацию по коллекции объектов и сообщает элементу управления о добавлении или удалении объектов из коллекции. *BindingSource* имеет методы навигации: ***MoveNext()***, ***MoveLast()***, ***MovePrevious()*** и ***MoveFirst()***. С помощью этих методов можно настроить свойства ***Position*** и ***Current*** соответствующим образом.

У каждого элемента управления (наследника класса Control) есть свойство ***DataBindings*** – коллекция экземпляров класса *Binding*, привязок к данным. Каждый экземпляр *Binding* обладает, как минимум, тремя свойствами: источник данных (data source), свойство (поле) источника данных и привязываемое свойство элемента управления (например, надпись на кнопке).

Для демонстрации техники простой привязки данных рассмотрим связывание текстовых полей TextBox, календаря (элемент управления DateTimePicker), текстового поля с маской (MaskedTextBox), флажка (ChekBox), списка (ListBox) с элементами данных объекта DataSet.

Создайте в проекте еще одну Windows-форму (рисунок 1). Добавьте программный код для выполнения простого связывания каждого из элементов управления с единственным полем базы данных.

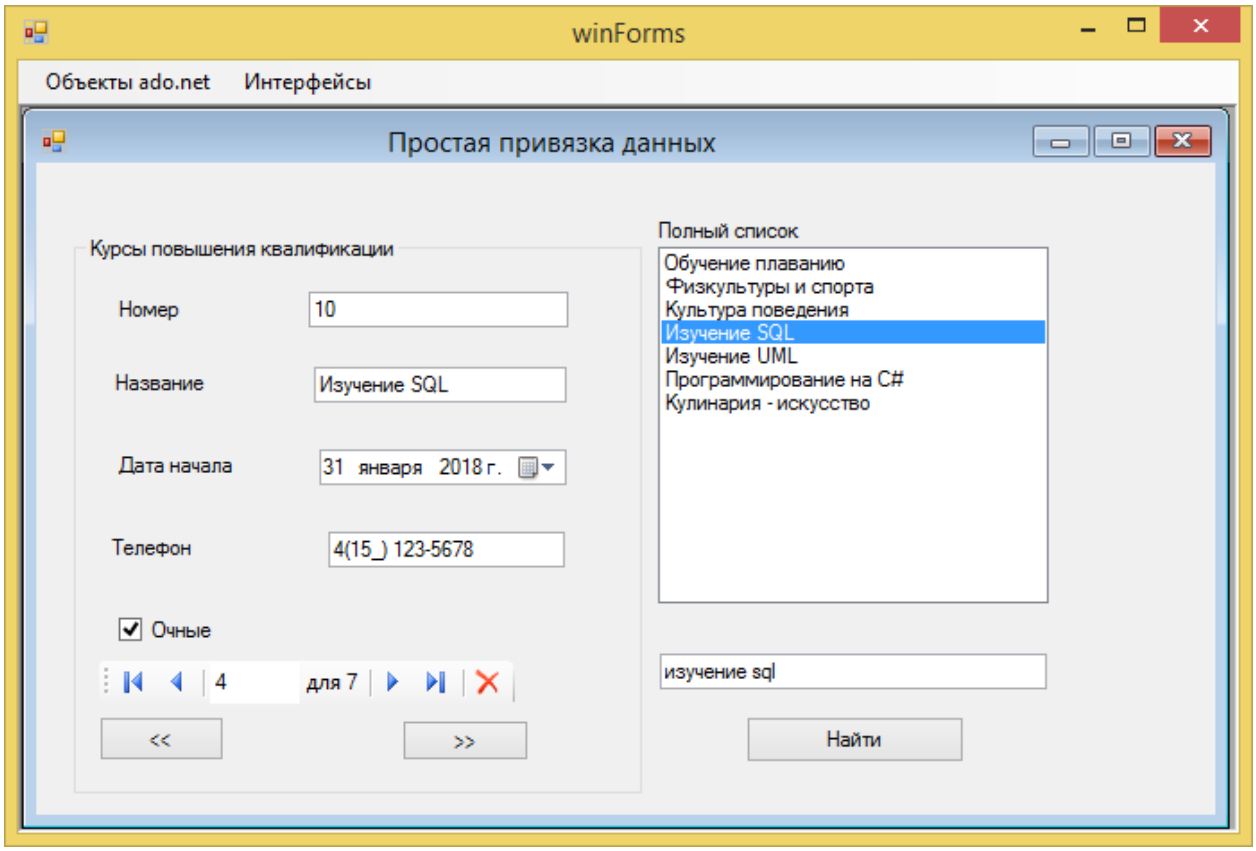
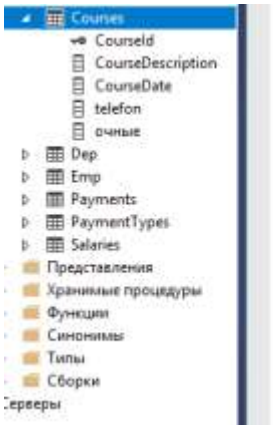


Рисунок 1 - Простая привязка данных к элементам формы




```
private void Form6_Load(object sender, EventArgs e)
{
 string str = ConfigurationManager.ConnectionStrings["Employees"].ConnectionString;
    cnn = new SqlConnection(str);
    da = new SqlDataAdapter("select * from Courses", cnn);
    da.Fill(ds, "Courses");
    bild = new SqlCommandBuilder(da);
    //Привязки элементов к данным
    bindingSource1.DataSource = ds.Tables[0];
    textBox1.DataBindings.Add("Text", bindingSource1, "CourseId");
    textBox2.DataBindings.Add("Text", bindingSource1, "CourseDescription");
    dateTimePicker1.DataBindings.Add("Value", bindingSource1, "CourseDate");
    maskedTextBox1.DataBindings.Add("Text", bindingSource1, "telefon");
    checkBox1.DataBindings.Add("Checked", bindingSource1, "очные");
    listBox1.DataSource = ds.Tables[0];
    listBox1.DisplayMember = "CourseDescription";
    listBox1.DataBindings.Add("Text", bindingSource1, "CourseDescription");
}
```

```
//Щелчок по элементу списка отображает полную запись
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    bindingSource1.Position = listBox1.SelectedIndex;
}
```

```
//Найти запись
private void button3_Click(object sender, EventArgs e)
{
    bindingSource1.Position = bindingSource1.Find("CourseDescription",
      textBox3.Text);
}
```

```
//Перемещение по записям
private void button1_Click(object sender, EventArgs e)
{
    bindingSource1.MovePrevious();
}

private void button2_Click(object sender, EventArgs e)
{
    bindingSource1.MoveNext();
}
```

Навигацию по записям bindingSource, добавление и удаление записей можно реализовать методами bindingSource или с помощью элемента bindingNavigator как на рисунке 1.

Привязка данных к элементу управления DataGridView в Windows Forms

Элемент управления *DataGridView* поддерживает стандартную модель привязки данных Windows Forms, допускающую привязку к разнообразным источникам данных. Однако в большинстве случаев выполняется привязка к компоненту *BindingSource*, который управляет деталями взаимодействия с источником данных. Компонент *BindingSource* может представлять любой источник данных Windows Forms и обеспечивает большую гибкость при выборе или изменении расположения данных.

Добавьте в новую форму 2 элемента отображения *DataGridView* и 2 элемента для связи *BindingSource* из панели инструментов. Выполните все необходимые связи между новыми элементами и реализуйте синхронное отображение записей в форме как показано на рисунке 2.

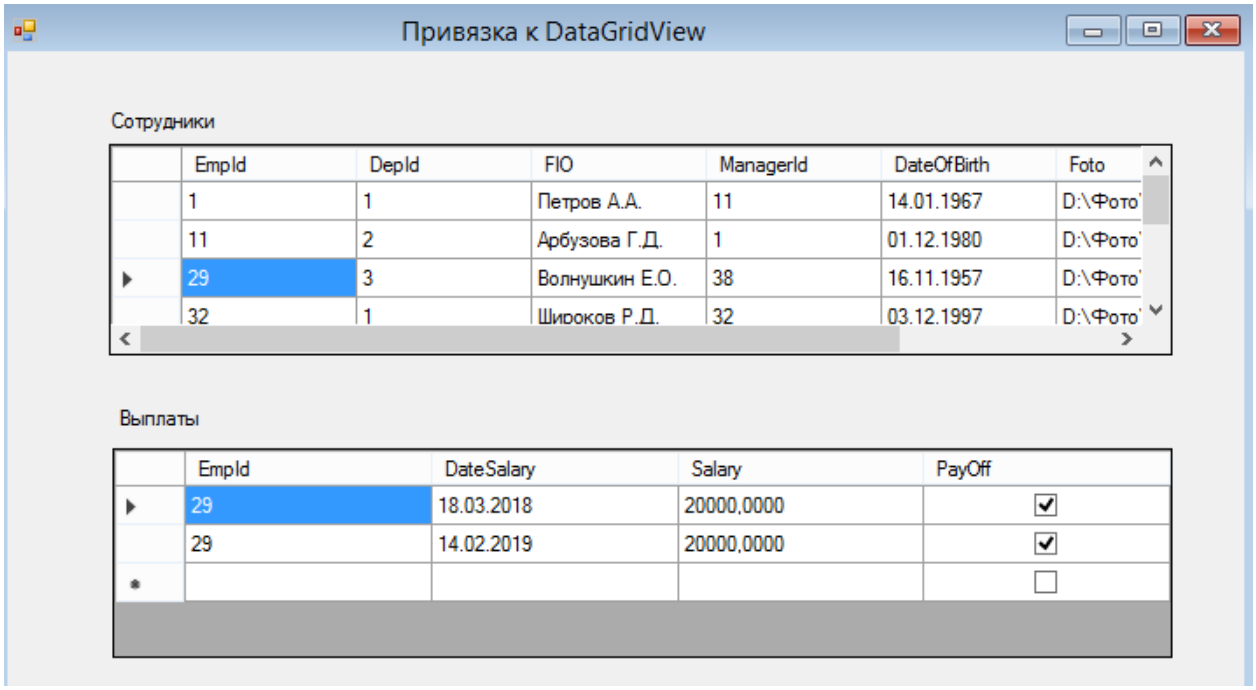


Рисунок 2 - Данные из двух связанных таблиц базы данных

```
    private void Form7_Load(object sender, EventArgs e)
    {
cnn = new SqlConnection(ConfigurationManager.ConnectionStrings["Employees"].ConnectionString);
        //Адаптеры данных
        daEmp = new SqlDataAdapter("select * from Emp", cnn);
        daSal = new SqlDataAdapter("select * from Salaries", cnn);
        daEmp.Fill(ds, "Emp");
        daSal.Fill(ds, "Sal");
        //Отношение между наборами данных в DataSet
ds.Relations.Add("E_S", ds.Tables["Emp"].Columns["EmpId"], ds.Tables["Sal"].Columns["EmpId"]);
        //Обеспечим возможность внесения изменений в таблицу Salaries
        bild = new SqlCommandBuilder(daSal);
        //Привязки элементов к данным DataGridView
        bind1.DataSource = ds.Tables["Emp"];
        dataGridView1.DataSource = bind1;
        bind2.DataSource = bind1;
        bind2.DataMember = "E_S";
        dataGridView2.DataSource = bind2;
    }
```

Привязка данных к элементу управления DetailsView в Web Form

Элемент *DetailsView* спроектирован для отображения отдельных записей в web-интерфейсе. Он размещает каждую часть информации в отдельной строке таблицы.

Элемент *DetailsView* может быть привязан к коллекции элементов. В этом случае он отображает первый элемент в группе. Он также позволяет перемещаться от одной записи к другой, используя страничные элементы управления, если установить свойство *AllowPaging* в *true*. Элементы управления страницами при этом можно конфигурировать, используя соответствующие свойства.

На рисунке 3 показан элемент *DetailsView*, привязанный к набору записей о курсах повышения квалификации:

```
protected void Page_Load(object sender, EventArgs e)
{
    SqlConnection cnn = new SqlConnection
      (WebConfigurationManager.ConnectionStrings
      ["Employees"].ConnectionString);
    SqlDataAdapter  da=new SqlDataAdapter("select * from Courses",cnn);
    DataSet ds = new DataSet();
    da.Fill(ds, "Courses");
    DetailsView1.DataSource = ds.Tables[0];
    DetailsView1.DataBind();
}
//Переход по страницам DetailsView
protected void DetailsView1_PageIndexChanging(object sender,
  DetailsViewPageEventArgs e)
{
    DetailsView1.PageIndex = e.NewPageIndex;
    DetailsView1.DataBind();
}
```

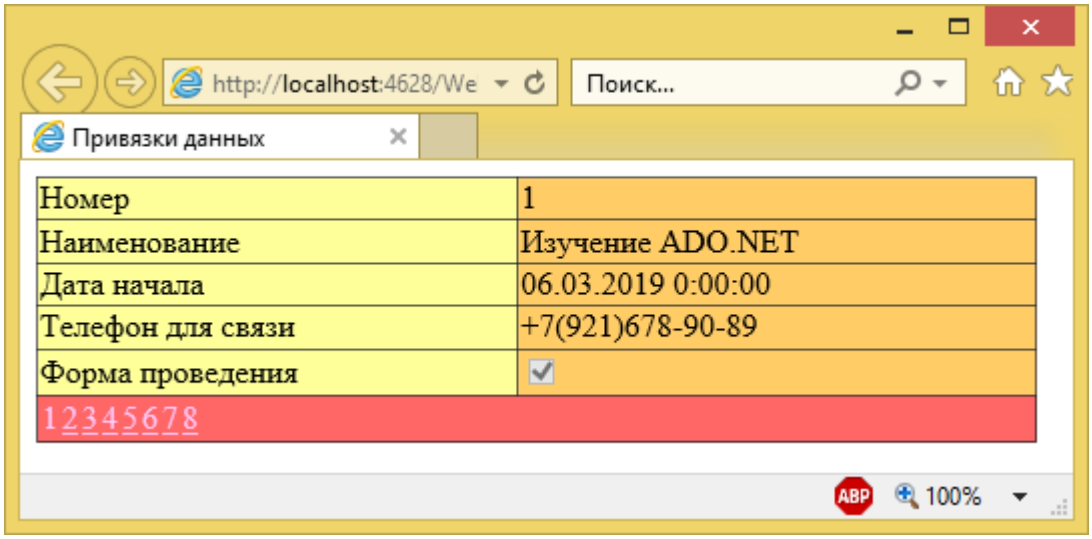


Рисунок 3 - Элемент DetailsView, привязанный к набору записей

Составная привязка данных в формах Windows

Простой привязкой данных (simple data binding) называется связывание элемента управления с единственным элементом данных.

Составная привязка данных (complex data binding) используется при связывании элемента управления сразу с несколькими элементами данных, например, связывание элемента управления «комбинированный список» (***comboBox***) с несколькими полями в источнике данных: с полем, отображаемым для пользователя, со скрытым значением, используемым при операции обновления записи в самой базе данных.

Добавьте в форму с привязкой данных к *DataGridView* элемент управления *comboBox* как на рисунке 4. В форму переместите еще 1 элемент ***bindingSource*** для синхронизации записей из связанных таблиц.

Внесите изменения в программный код для обработки события ***Form_Load***, в рамках которого будет осуществляться загрузка ***DataSet*** и привязка его таблиц к элементам формы ***comboBox*** и ***dataGridView***.

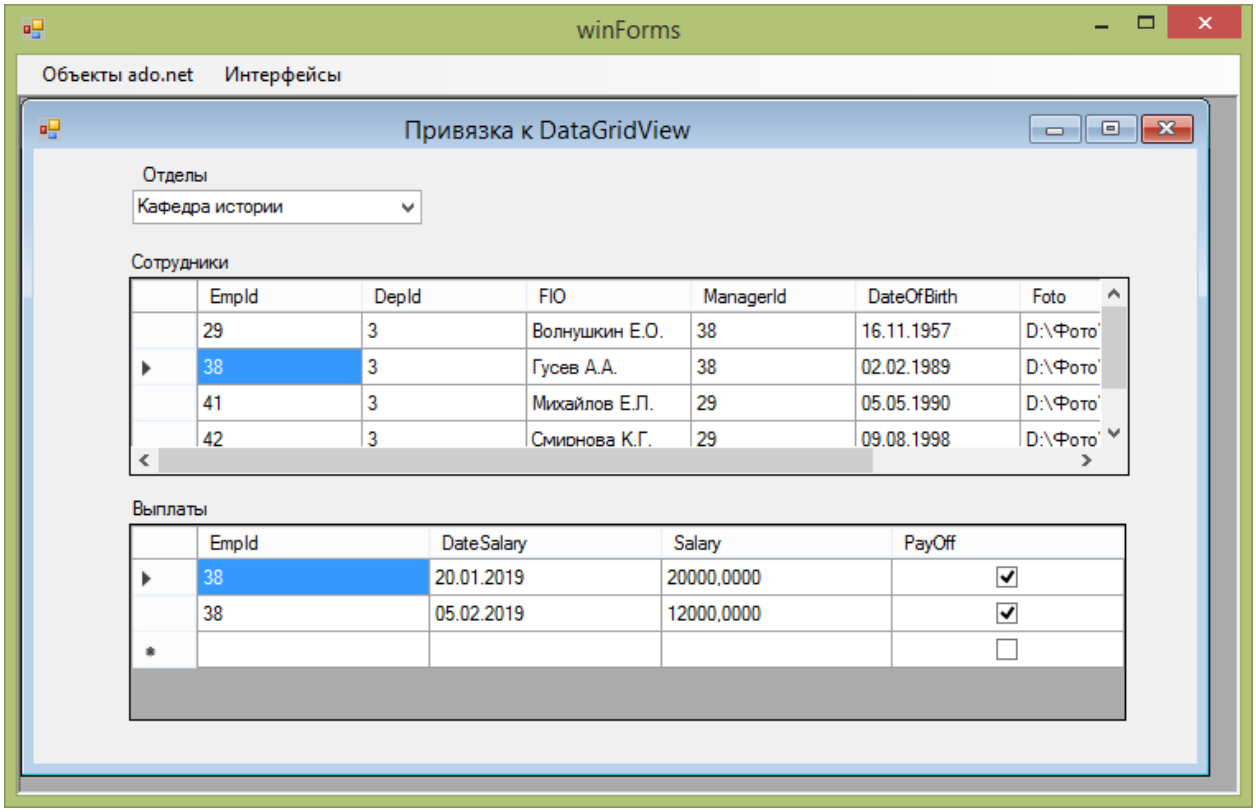


Рисунок 4 - Составная (complex) привязка данных к элементу формы comboBox

```
    private void Form7_Load(object sender, EventArgs e)
    {
cnn = new SqlConnection(ConfigurationManager.ConnectionStrings
  ["Employees"].ConnectionString);
        //Адаптеры данных
        daDep = new SqlDataAdapter("select DepId, Description from Dep",cnn);
        daEmp = new SqlDataAdapter("select * from Emp", cnn);
        daSal = new SqlDataAdapter("select * from Salaries", cnn);
        daDep.Fill(ds, "Dep");
        daEmp.Fill(ds, "Emp");
        daSal.Fill(ds, "Sal");
        //Отношение между наборами данных в DataSet
        ds.Relations.Add("D_E",ds.Tables["Dep"].Columns["DepId"],ds.Tables["Emp"].Columns
          ["DepId"]);
ds.Relations.Add("E_S", ds.Tables["Emp"].Columns["EmpId"], ds.Tables["Sal"].Columns
  ["EmpId"]);
        //Обеспечим возможность внесения изменений в таблицу Salaries
        bild = new SqlCommandBuilder(daSal);
        //Привязки элементов к данным DataGridView
        bind1.DataSource = ds.Tables["Dep"];
        //Составная привязка
        comboBox1.DataSource = bind1;
        comboBox1.DisplayMember = "Description";
        comboBox1.ValueMember = "DepId";
        //Простая привязка
        bind2.DataSource = bind1;
        bind2.DataMember = "D_E";
        dataGridView1.DataSource = bind2;
        bind3.DataSource = bind2;
        bind3.DataMember = "E_S";
        dataGridView2.DataSource = bind3;
    }
```

Самостоятельно

1) Разработайте в windows-форме привязку данных к элементам интерфейса. Желательно, чтобы данные в таблице имели разнообразные типы, а в интерфейсе к ним были привязаны различные элементы управления. Используйте самостоятельно метод ***RemoveCurrent()*** объекта *BindinSource* для удаления текущей записи формы. Организуйте сохранение изменений, выполненных в форме, в базу данных.
2) Реализуйте отображение связанных и синхронизированных данных двух таблиц базы данных. Сохранение изменений можно реализовать для одной таблицы.

3*) Использование привязки данных к элементам web-формы ***можно*** выполнить для элемента DetailsView.

4) Разработайте интерфейс (можно windows или web) для реализации комплексной привязки данных по образцу рисунка 4.
5) Не забудьте реализовать возможность сохранения изменений в одной из таблиц. В рассмотренных примерах изменения возможны в таблице Salaries.