Часть II

РЕАЛИЗАЦИЯ И ЭКСПЛУАТАЦИЯ БАЗ ДАННЫХ

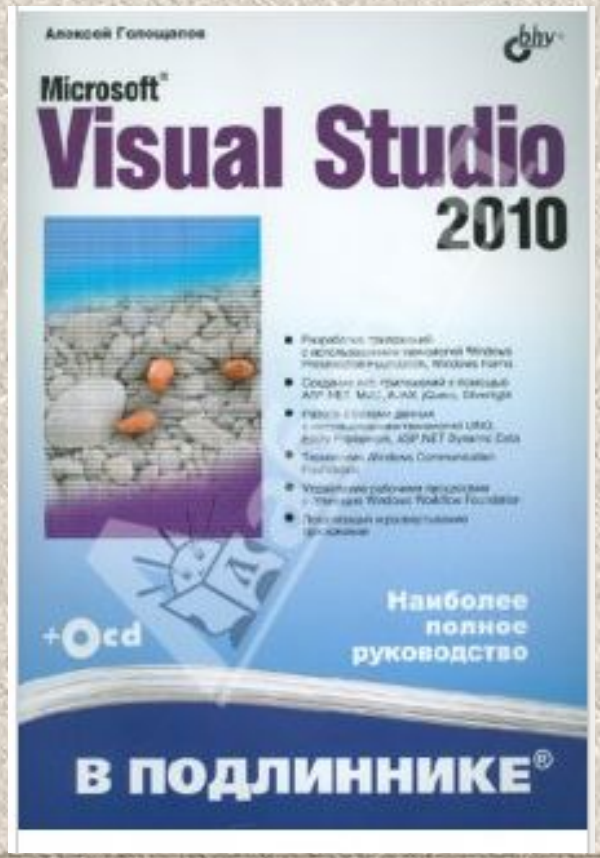
Раздел V

ПРОГРАММНЫЙ ИНТЕРФЕЙС ДОСТУПА К ДАННЫМ

Лекция 18

Технология доступа к данным в среде VISUAL STUDIO 2010

Рекомендуемые источники



ISBN: 978-5-9775-0617-5

Автор: Голощапов Алексей Леонидович

Редактор: Кондукова Екатерина

Издательство: BHV, 2011 г.

Серия: В подлиннике

Страниц: 544 (Газетная)

Тип обложки: обл - мягкий переплет (крепление скрепкой или клеем)

Иллюстрации: Черно-белые

Масса: 488 г

Размеры: 232x166x23 мм

Этапы создания клиентского приложения

Создание интерфейса клиентского приложения в Visual Studio происходит в несколько этапов:

- Создаётся проект;
- В проекте создаются объекты связи, которые подключаются к файлу данных;
- Создаются формы;
- Создаются отчёты.

§1 Технологии доступа к данным и Объекты связи

ODBC

Низкоуровневая технология

ODBC (Open Database Connectivity) — открытый интерфейс баз данных.

Необходимость создания ODBC появилась вследствие того, что каждая фирма — разработчик СУБД использовала свой диалект SQL, что делало невозможным обмен данными между двумя БД различных форматов. Поэтому вначале был разработан общий стандарт на SQL, получивший название CLI (Common Language Interface). Затем каждая фирма разрабатывала драйвер перевода своего диалекта SQL в CLI и наоборот.

ODBC предназначена для обеспечения возможности взаимосвязи между различными SQL-совместимыми БД.

В архитектуре ODBC используется один ODBC Driver Manager и несколько ODBC-драйверов, отвечающих за реализацию особенностей доступа к каждой отдельной СУБД.

Преимущества:

- простота разработки приложения;
- технология ODBC позволяет создавать распределенные гетерогенные приложения без учета конкретных СУБД, т.е. приложение становится независимым от СУБД.
- данные в БД могут быть представлены в любом виде и формате (электронные таблицы, документы в rtf- формате, почтовые системы и т.д.).

Недостатки:

- снижение скорости доступа к данным из-за необходимости трансляции запросов;
- увеличение время обработки запросов из-за введения дополнительного программного слоя;
- необходимы предварительная инсталляция и настройка ODBC-драйвера (указание драйвера СУБД, сетевого пути к серверу, базы данных и т.д.) на каждом рабочем месте. Параметры этой настройки являются статическими, т.е. приложение изменить их самостоятельно не может;
- предоставляет доступ только к реляционным SQL-ориентированным БД. OLE DB.

OLE DB

OLE DB (Object Linking and Embedding Data Base) — технология, предоставляющее решение обеспечения COM-приложениям доступ данным независимо от типа источника данных.

В технологии OLE DB используется механизм провайдеров, под которыми понимают поставщиков данных.

Провайдер представляет собой компонент COM, позволяющий принимать вызовы OLE DB и выполнять все необходимое для обработки запроса к источнику данных. Провайдер возвращает запрашиваемый объект — обычно это данные в табличном виде.

Кроме поставщика данных имеются также сервис провайдеры, реализующие самые различные сервисные функции.

Технологии ODBC и OLE DB считаются хорошими интерфейсами передачи данных, но как программные интерфейсы имеют много ограничений, поскольку являются низкоуровневыми.

DAO, ADO

Для снятия ограничений ODBC и OLE DB были предложены технологии DAO и ADO.

Данные технологии представляют собой высокоуровневые объектные модели (библиотеки функций) и создают еще один уровень абстракции между приложением и функциями ODBC и OLE DB.

DAO (Data Access Objects) можно использовать для операций с источниками данных ODBC двумя способами: через Microsoft Jet или новую технологию *ODBCDirect.*

Технология DAO предназначена преимущественно для создания БД с помощью СУБД MS Access, т.к. кроме замены функций ODBC она осуществляет также прямой доступ к функциям ядра MS Jet базы данных Access.

ADO (ActiveX Data Objects) технология ADO представляет иерархическую модель объектов для доступа к различным OLE DB- провайдерам данных. Объектная модель ADO включает объекты, обеспечивающие соединение с провайдером данных, создание SQL-запросов к данным и т. д.

Модель объекта не содержит таблиц, среды. Здесь основными объектами являются:

- объект Набор данных;
- объект Соединение, создающий связь с провайдером данных;
- объект Команда — выполнение процедуры.

Особенностью технологии ADO является возможность ее использования в Интернет/Интранет-приложениях для доступа к различным источникам данных.

В целом технологию ADO можно охарактеризовать как наиболее современную технологию разработки приложении для работы с распределенными БД по технологии клиент-сервер.

BDE

BDE (Borland Data Engine) — технология фирмы Borland.

Данная технология реализована в виде динамически подключаемых библиотек и имеет достаточно развитый интерфейс прикладных программ, названный IDAPI (Integrated Database Application Program Interface).

Этот интерфейс представляет собой набор функций для работы с базами данных

Является некоторым аналогом ODBC. Как и ODBC технология BDE имеет набор драйверов для работы с различными СУБД. Если собственного драйвера для доступа к некоторой СУБД в BDE нет, то используется драйвер доступа к ODBC.

JDBC

JDBC (Java Data Base Connectivity) — мобильный интерфейс к базам данных на платформе Java. Это интерфейс прикладного программирования для выполнения SQL-запросов к базам данных из программ, написанных на платформенно-независимом языке Java, позволяющем создавать как самостоятельные приложения, так и аплеты, встраиваемые в Web-страницы

Объекты связи

Объекты связи - это объекты проекта, осуществляющие обмен информацией между интерфейсом БД и файлом данных.

Объекты связи всегда находятся на клиентской машине. Они осуществляют доступ к файлам данных, передавая информацию в интерфейс БД, и содержат внутри себя запросы, выполнения на стороне клиента.

Замечание: Объекты связи также могут ограничивать доступ к информации и осуществлять защиту информации, хотя для защиты информации и ограничения доступа лучше использовать сам сервер.

Технологии в объектах связи

Существует три технологии используемых в объектах связи:

- технология ADO;
- технология RDO;
- технология ADO.Net.

Технология ADO

ADO (от англ. *ActiveX Data Objects* — «объекты данных ActiveX») — интерфейс программирования приложений для доступа к данным, разработанный компанией Microsoft (MS Access, MS SQL Server) и основанный на технологии компонентов ActiveX.
ADO позволяет представлять данные из разнообразных источников (реляционных баз данных, текстовых файлов и т. д.) в объектно-ориентированном виде.

ADO является более старой технологией.

Её суть заключается в следующем: подключение к конкретной таблице или запросу, осуществляется через отдельный объект связи, т.е. все настройки и средства для работы с данными хранятся внутри конкретного объекта связи и были заложены туда при его проектировании.

Технология RDO

RDO (сокр. от англ. Remote Data Objects) — технология доступа к базам данных компании Microsoft. Технология RDO появилась в 1995 году.
RDO позиционировалась как технология более простая чем прямое использование вызовов ODBC и в то же время более эффективная чем технология DAO. RDO была ориентирована на обработку данных на стороне сервера БД (такого как MS SQL Server, Oracle и т.д.) в отличие от DAO ориентированной в основном на обработку данных на стороне клиента.

Согласно технологии **RDO** файлы данных рассматриваются в качестве устройств, т.е. для работ с БД необходим драйвер.

Объект связи, работающий по технологии **RDO**, при работе с файлом данных сначала обращается к драйверу БД, который в свою очередь обращается к файлу данных.

Технология ADO.Net

ADO.NET (ActiveX Data Objects .NET) — основная модель доступа к данным для приложений, основанных на Microsoft .NET. Не является развитием более ранней технологии ADO, а представляет собой совершенно самостоятельную технологию. Компоненты ADO.NET входят в поставку оболочки .NET Framework; таким образом, ADO.NET является одной из главных составных частей .NET.

Технология **ADO.Net** является смесью технологий **ADO** и **RDO**. Объекты связи работающие по этой технологии работают аналогично объектам работающим по технологии **ADO**, однако, объекты связи входят в состав пакета Microsoft Net Framework, и автоматически обновляются вместе с этим пакетом.

Плюсы и минусы технологий

ADO

+ независимость от драйверов БД, установленных в операционной системе
+ простое программирование

- невозможность работать с новыми типами БД
- невозможность обновлять список поддерживаемых БД

RDO

+ возможность работать с современными БД
+ возможность добавлять новые виды БД

- зависимость от драйверов, установленных в системе
- более сложное программирование

ADO.Net

+ возможность работать с современными БД
+ возможность добавлять новые виды БД

- зависимость от пакета Microsoft Net Framework
- более сложное программирование

Замечание: Мы можем создавать динамические запросы и запросы, выполненные на стороне сервера только в технологии RDO и ADO.Net.

§2 Подключение проекта к файлу БД

Мастер подключений

В Visual Studio подключение проекта к файлу БД можно произвести двумя способами:

- при помощи мастера подключений и вручную,
- создавая объекты связи и настраивая их свойства.

Начнем рассмотрение создания подключения с помощью мастера.

Создание подключения состоит из создания следующих объектов:

DataSet (Набор данных)	обеспечивает подключение формы к конкретной БД на сервере
BindingSource (Источник связи)	обеспечивает подключение к конкретной таблице, а также позволяет управлять таблицей
TableAdapter (Адаптер таблиц)	обеспечивает передачу данных с формы в таблицу и наоборот
TableAdapterManager (Менеджер адаптера таблиц)	управляет работой объекта **TableAdapter**
BindingNavigator (Панель управления таблицей)	панель с кнопками управления таблицей, расположенная в верхней части формы

Можно создать и подключить все эти объекты вручную, но удобнее воспользоваться мастером.

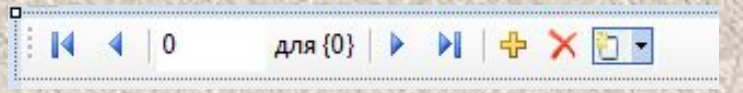
Работа с мастером подключений состоит из нескольких этапов:

1. Запуск мастера.
2. Выбор типа источника данных: БД, сетевой источник или объект.
3. Настройка строки подключения **"Connection String"**. Настройка заключается в выборе вида БД (либо Access, либо SQL Server), а также в выборе сервера и файла данных. В случае необходимости можно задать логин и пароль.
4. Сохранение строки подключения. При ее сохранении можно менять параметры подключения без использования Visual Studio. Но при сохранении строки подключения в файл велика вероятность несанкционированного подключения к БД.
5. Выбор таблиц или запросов включённых в соединение. Также можно выбрать их отдельные поля.
6. Завершение работы мастера подключений.

Настройка связи подключение вручную

В Visual Studio можно создавать объекты связи вручную и их настраивать.

Для связи Visual Studio использует три объекта связи, причем они работают все вместе, плюс к этому был и существует объект **BindingNavigatior** (Панель навигации) - эта панель обеспечивает полное управление источником данных (добавление, удаление, перемещение по записям).



Создание и настройка соответствующих объектов связи

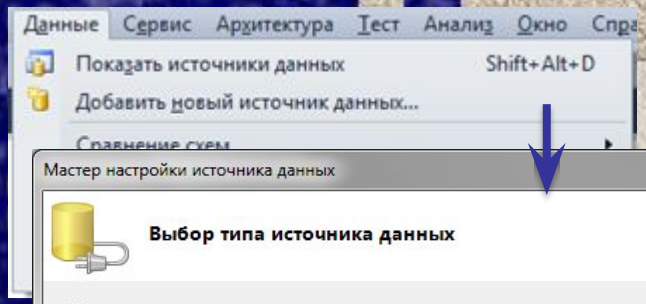
1. Создание подключения начинается с создания объекта **DataSet**.

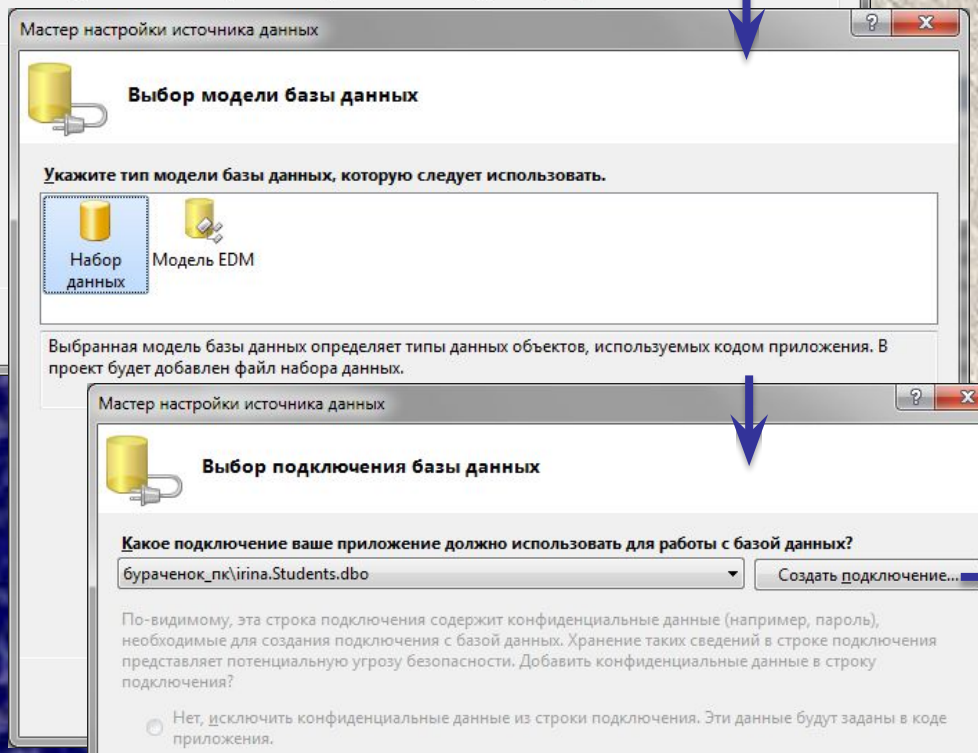
Объект **DataSet** не может сам подключится к источнику данных перед его созданием необходимо настроить **"DataSources"** (оконное меню **Data\Add Data Sources**).

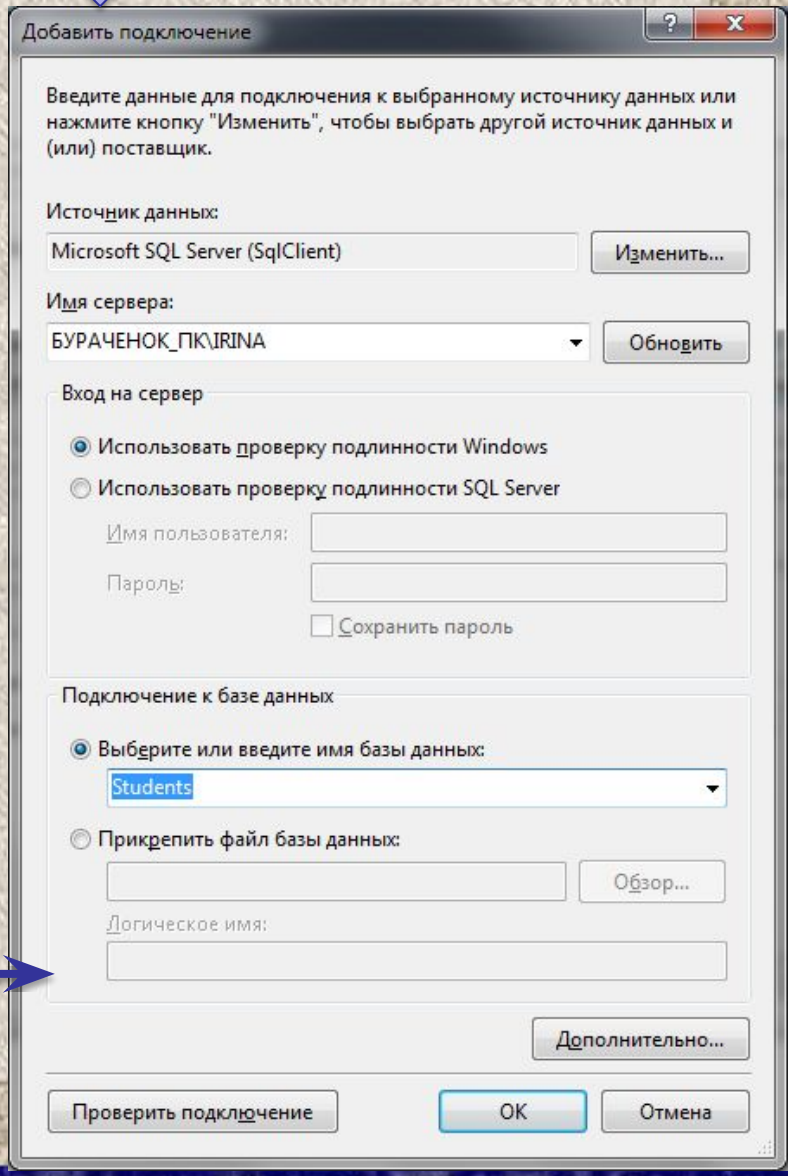
После создания объекта **DataSet** появляется окно **"Add Data Set"**. В нем необходимо в выпадающем списке **"Typed Data Set"** выбрать источник данных из **"Data Sources"**.

После выбора источника данных в списке **"TypedDataSet"** появится строка **Windows Application <имя источника>**. После этого в окне можно нажать кнопку **"Ok"**. Имя источника данных будет записана в свойство **DataSetName** объекта **DataSet**.

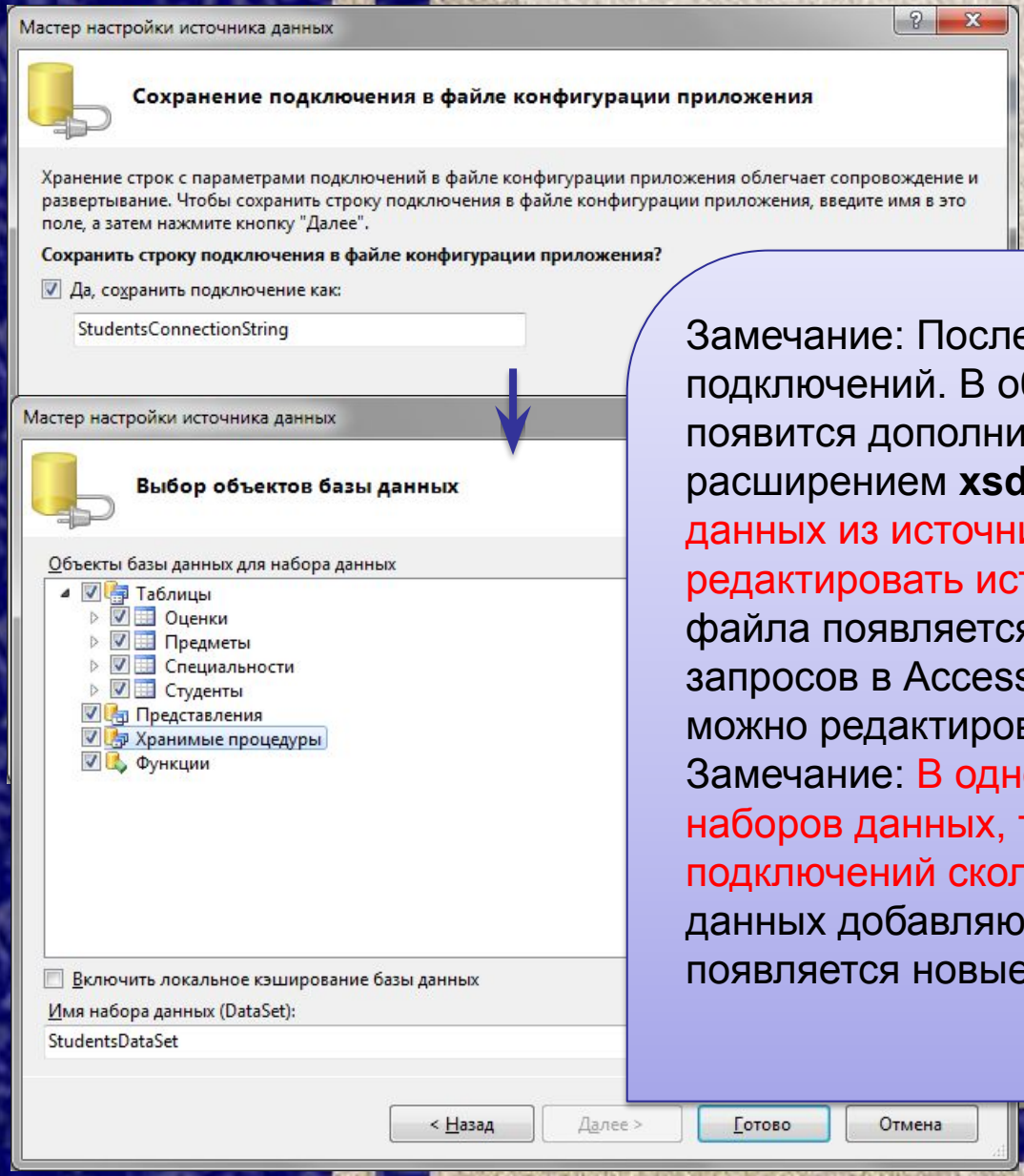
Пример подключения к БД





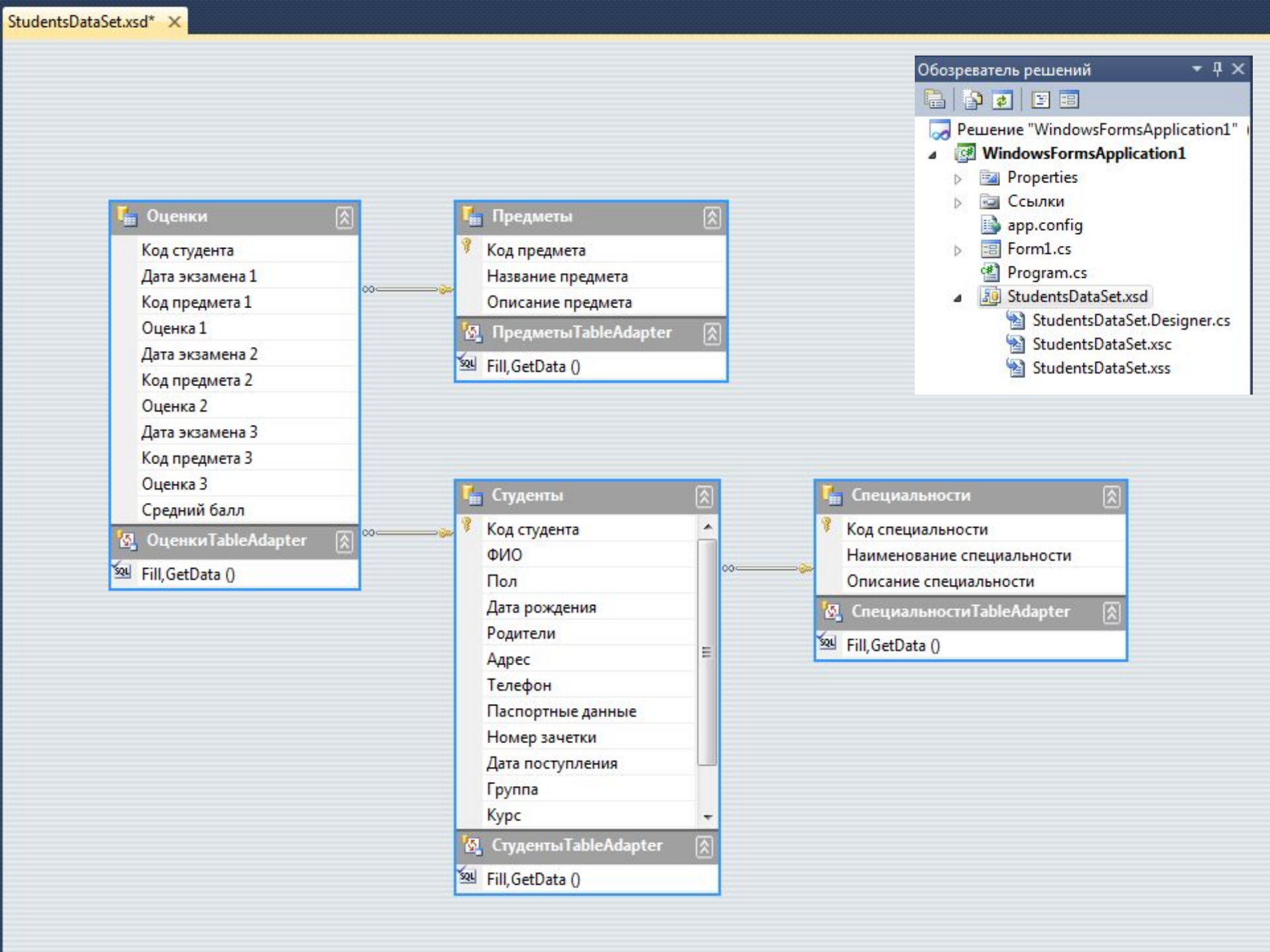


Файл набора данных



Замечание: После окончания работы мастера подключений. В обозревателе в "**SolutionExplorer**" появится дополнительный файл набора данных с расширением **xsd**. Этот файл содержит в себе схему данных из источника данных, а также позволяет редактировать источник данных (при открытии этого файла появляется окно похожее на конструктор запросов в Access или SQL Server), в этом окне также можно редактировать поля таблиц.

Замечание: В одном проекте может быть несколько наборов данных, то есть можно запускать мастер подключений сколько угодно раз. Новые наборы данных добавляются на вкладку "**DataSources**" и появляется новые данные с расширением **xsd**.


StudentsDataSet.xsd*
Обозреватель решений
Решение "WindowsFormsApplication1"
WindowsFormsApplication1
Properties
Ссылки
app.config
Form1.cs
Program.cs
StudentsDataSet.xsd
StudentsDataSet.Designer.cs
StudentsDataSet.xsc
StudentsDataSet.xss
Оценки
Код студента
Дата экзамена 1
Код предмета 1
Оценка 1
Дата экзамена 2
Код предмета 2
Оценка 2
Дата экзамена 3
Код предмета 3
Оценка 3
Средний балл
ОценкиTableAdapter
Fill,GetData ()
Предметы
Код предмета
Название предмета
Описание предмета
ПредметыTableAdapter
Fill,GetData ()
Студенты
Код студента
ФИО
Пол
Дата рождения
Родители
Адрес
Телефон
Паспортные данные
Номер зачетки
Дата поступления
Группа
Курс
СтудентыTableAdapter
Fill,GetData ()
Специальности
Код специальности
Наименование специальности
Описание специальности
СпециальностиTableAdapter
Fill,GetData ()

Создание объекта BindingSource

2. После создания объекта **DataSet** создается объект **BindingSource**, он позволяет подключится к таблицам, запросам и фильтрам из файла данных. После его создания необходимо настроить следующие свойства:

DataSource - указанный объект **DataSet**;

DataMember - указывает таблицу, запрос или фильтр, которые будут отображаться на форме.

Следующие свойства необязательны для настройки:

Filter - свойство для фильтрации данных, в нем записывается условие отбора для какого-то поля;

Sort - сортировка информации

Allow New - позволяет добавлять новые записи.

Создание объекта TableAdaper и BindingNavigatior

3. После добавления **DataSet** и **BindingSource** автоматически будет добавлен объект **TableAdapter**.

После чего уже можно добавлять объекты для отображения данных, однако, при этом нельзя будет управлять информацией.

4. Для управления источником данных создаётся объект **BindingNavigatior**.
Затем его необходимо подключить к объекту **BindingSource**. Для этого в свойстве **BindingSource** объекта **BindingNavigatior** необходимо указать созданный ранее объект **BindingSource**.

Настройка вида панели навигации

Затем можно настроить внешний вид панели навигации при помощи следующих свойств:

AddNewItem - отображает кнопку для добавления новой записи;

DeleteItem - отображает кнопку для удаления текущей записи;

AddNextItem - отображает кнопку для добавления новой записи после текущей;

MoveFirstItem - отображает кнопку для перехода к первой записи;

MoveNextItem - отображает кнопку для перехода к следующей записи;

MovePreviousItem - отображает кнопку для перехода к предыдущей записи;

MoveLastItem - отображает кнопку для перехода к последней записи;

CountItem - отображает общее количество записей;

PositionItem - отображает номер текущей записи.

§2 Интерфейс информационных систем. Создание интерфейса пользователя

Интерфейс информационных систем

В системах построенных по технологии клиент-сервер существует два вида интерфейса:

- Интерфейс, реализуемый при помощи клиентского приложения;
- Web -интерфейс.

Интерфейс, реализуемый при помощи клиентского приложения

Интерфейс, реализуемый при помощи клиентского приложения - это компьютерная программа, устанавливаемая на клиентские компьютеры, предназначенная для работы с файлами данных через сеть.

Основными элементами клиентских приложений являются формы (окно программы) и отчёты.

Элементы управления на форме называется **объектами**.

Каждый объект обладает своим набором свойств, событий и методов.

Объекты форм

В БД все объекты форм делятся на два класса:

- **Объекты управления** - объекты, осуществляющие управление БД (Например: *Кнопка* или *Выпадающий список*);
- **Объекты для отображения информации** - элементы, отображающие содержимое таблиц, запросов или фильтров, позволяющие добавлять изменять и удалять информацию, и проводить ее анализ.

Формы в клиентском приложении

Все формы в клиентском приложении делятся на три группы:

Формы для работы с данными - формы, содержащие как объекты управления, так и объекты просмотра данных. Такие формы предназначены для отображения, изменения, удаления и анализа данных;

Кнопочные формы - формы, содержащие только объекты управления, предназначаются для открытия всех других форм.

Замечание: Кнопочная форма, которая появляется первой после запуска программы, называется, главной кнопочной формой.

Информационные и служебные формы - формы, содержащие только элементы управления, предназначены для отображения служебной информации (справки), несвязанной с таблицами, запросами и фильтрами, либо для выполнения служебных операций не связанных с данными (Например: форма с калькулятором)

Виды дизайна форм

Замечание: Существует два вида дизайна форм:

- Ленточные формы - формы, выводящие информацию по одной записи;
- Табличные формы - формы выводящие информацию в виде таблицы.

Замечание: Объекты связи используются только в клиентском интерфейсе. В web-интерфейса функции объекта связи выполняет сервер.

Основой web-интерфейса являются страницы (файл с расширенным htm или html). Работа со страницами осуществляется с помощью программы - браузера. Изначально страницы находятся на сервере, пользователь сначала загружает их на свой компьютер с сервера, а затем с помощью страниц пользователь работает с файлом данных.

Замечание: В web-интерфейсе отсутствуют отчёты, их роль выполняют сами страницы.

§2.1 Создание интерфейса пользователя
Создание ленточной формы

Создание интерфейса при помощи окна "DataSources"

Visual Studio позволяет создавать не сложный интерфейс БД, без помощи панели объектов и окна свойств, лишь используя окно **"DataSources"**.

В окне **"DataSources"** после подключения источника данных отображается все таблицы, запросы, фильтры данных и их поля. (Можно перетаскивать источники данных, соответственно таблицы, запросы, фильтры прямо из окна **"DataSources"** на форму. При перетаскивании можно выбирать для каждого поля источника данных объект, который будет отображать его содержимое.)

Замечание: Таким способом можно создавать только определённые объекты для отображения данных поля, и набор этих объектов зависит от типа данных поля.

Создание объектов для отображения данных перетаскиванием

Создание объектов для отображения данных перетаскиванием состоит из двух шагов:

1. Для каждого поля таблицы, запроса, или фильтра выбирается объект, который будет отображать его содержимое. Для этого необходимо щелкнуть мышью по полю в окне **"DataSources"**, рядом с именем поля появится кнопка, со стрелкой, щелкнув мышью по стрелке, отобразится выпадающее меню с объектами, которые могут отображать информацию, содержащуюся в поле.

- Для полей стандартными объектами являются: **TextBox**, **ComboBox**, **Label**, **LinkLabel**, **ListBox**.
- Для полей типа данных **ДатаВремя (DateTime)** возможно использования объекта **DataTimePicker**.
- Для полей логических типов данных возможно использование объекта **CheckBox**.

Создание объектов для отображения данных перетаскиванием

- Для отображения таблиц, запросов или фильтров целиком возможно два варианта отображения:
  - при помощи объекта **DataGridView** - информация из таблицы, запроса или фильтра отображается в виде таблицы;
  - **DetalledView** - отображение всех полей источника данных в **TextBox** по отдельности.

Замечание: В выпадающем меню с вариантами выбора объектов имеется пункт **"Customize"** (Настройки), который позволяет выбрать дополнительные допустимые объекты для отображения информации.

Создание объектов для отображения данных перетаскиванием

2. После выбора объектов для отображения необходимо их поместить на форму, перетаскивая мышью с панели "**DataSources**" в нужное место на форме.

Замечание: При помещении первого объекта на форму на ней автоматически создаются объекты для связей с файлом данных и объекты по навигациям по источникам данных (**DataSet**, **BindingSource**, **TableAdapter**, **BindingNavigator**).

Замечание: По умолчанию панель навигации располагается в верхней части формы. Эту панель можно прикрепить около различных краев формы. Для этого необходимо воспользоваться меню действий объектов. Чтобы вызвать это меню, необходимо выделить объект. В его правом верхнем углу появится кнопка (квадратик со стрелочкой), при нажатии этой кнопки появляется выпадающее меню с настройками и действия с объектом. Например, *чтобы поменять местоположение навигации панели - надо в этом меню выбрать настройку **Docking***.

Замечание: При перетаскивании на форму полей источников данных автоматически создаются подписи к ним (**Label**).

Подключение объектов к источнику данных при помощи окна свойств

Visual Studio позволяет подключать источники данных к объектам без использования перетаскивания, то есть вручную, с использованием панели свойств.

Для этого на форму помещается объект, который будет подключаться к источнику данных.

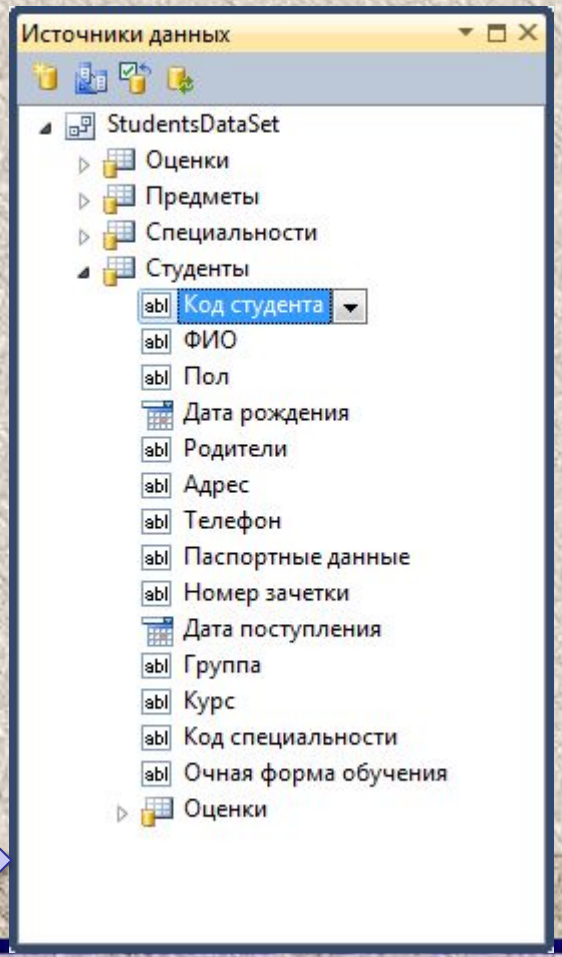
Его выделяют, затем на панели свойств разворачивается группа свойств **"DataBindings"** она содержит два свойства:

- **Text** - определяет таблицу, запрос или фильтр, из которого выводятся данные в объект.
- **Tag** - определяет поле, выбранного в свойстве **Text** источника данных, которое отображается в объекте.

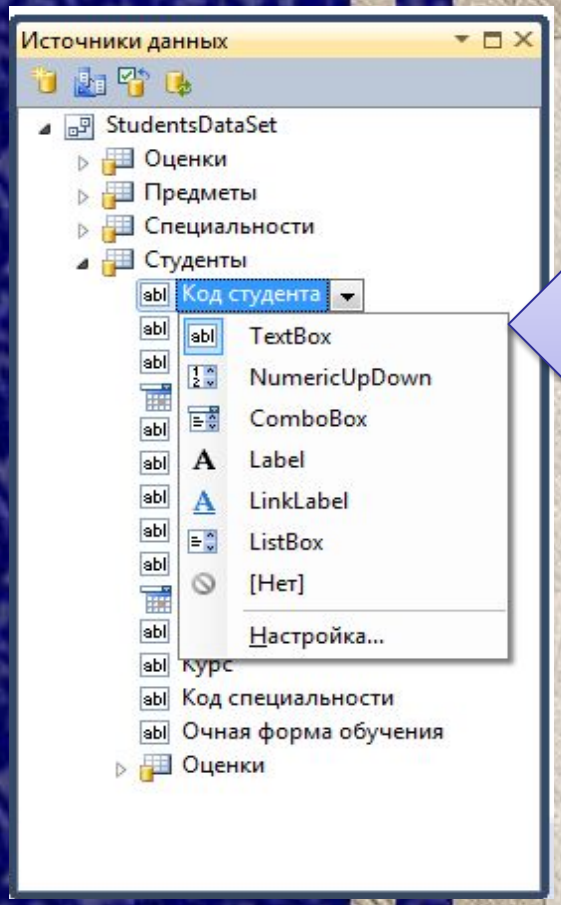
Источники данных

Открыть панель **"Источники данных"** (**Data Sources**) можно, щелкнув по ее вкладке в правой части окна среды разработки

На панели **"Источники данных"** отобразите поля необходимой таблицы щелкнув по значку , расположенному слева от имени таблицы.



Замечание: Под полями таблицы **"Студенты"** в виде подтаблицы располагается таблица **"Оценки"** . Подтаблица показывает, что таблица **"Студенты"** является вторичной по отношению к таблице специальности.

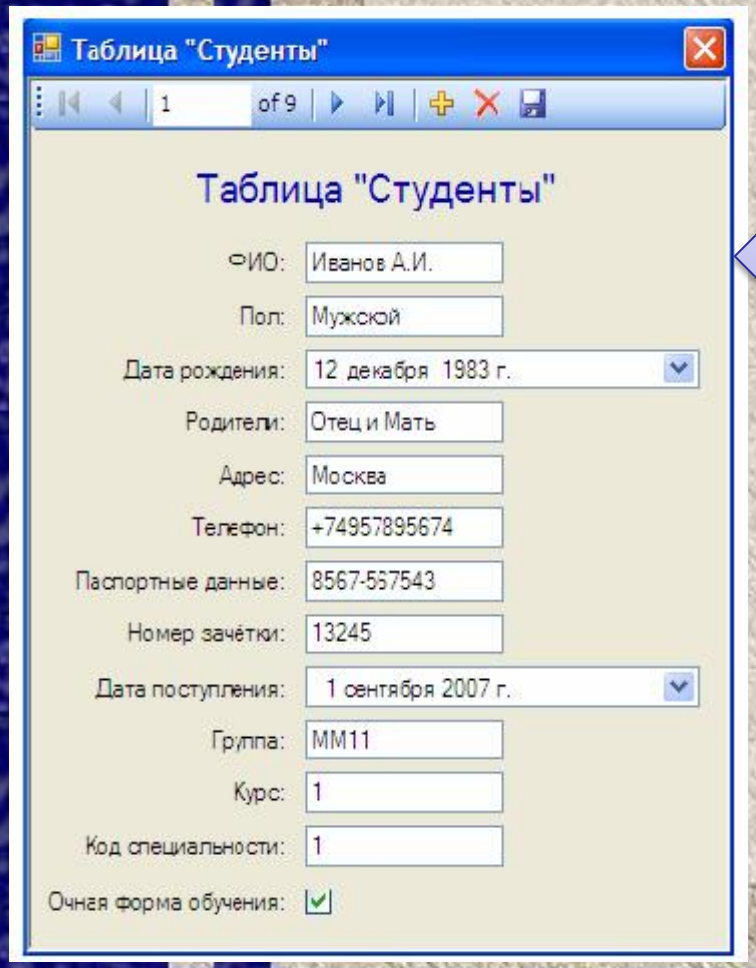


Замечание: При выделении, какого либо поля таблицы, оно будет отображаться в виде выпадающего списка, позволяющего выбирать объект, отображающий содержимое выделенного поля

Замечание: Мы не должны помещать поле **"Код специальности"** на форму, так как данное поле является первичным полем связи и заполняется автоматически. Конечный пользователь не должен видеть такие поля.

Замечание: После перетаскивания полей с панели **"Источники данных"** на форму в верхней части формы должна появится *навигационная панель*, а в нижней части рабочей области среды разработки появится *пять невидимых объектов.* Эти объекты предназначены для связи нашей формы с таблицей **"Студенты"**, расположенной на сервере.

Пример простой ленточной формы для работы с данными



Обратите внимание на объекты, отображающие поля "Дата рождения", "Дата поступления" и "Очная форма обучения".

§2.1.2 Стандартные объекты для отображения данных. Программное управление информационной системой

Стандартные объекты для отображения данных

Способ создания объектов для отображения данных описанный ранее, позволяет создавать только ограниченный набор объектов. Однако, Visual Studio позволяет подключать источники данных практически к любому объекту, который может быть создан на форме. Это можно сделать при помощи перетаскивания поля источника данных из окна "**Data Sources**" на объект на форме.

Операция состоит из двух шагов:

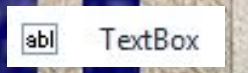
- При помощи панели объектов (слева) на форме создается какой-то объект. Объекты несвязанные с источником данных называют *несвязанными объектами*.
- Вновь созданный объект связывается с источником данных. Для этого на объект нужно перетащить поле таблицы запроса или фильтра из окна "**DataSources**".

Замечание: При перетаскивании поля из окна "**Data Sources**" необходимо учитывать его тип данных. Объект на форме должен поддерживать тип данных подключаемого к нему поля.

Замечание: В случае подключения объекта к источнику данных, способом, описанным выше, подпись к объекту не создаётся автоматически и её надо создавать вручную с помощью объекта **Label**.

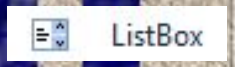
Наиболее часто в БД используются следующие объекты для отображения информации:

- Текстовое поле (**TextBox**)
- Надпись (**Label**)
- Надпись со ссылкой (**LinkLabel**)
- Календарь (**DataPicker**)
- Переключатель (**CheckBox**)
- Таблица (**DataGridView**)
- Список (**ListBox**)
- Выпадающий список (**ComboBox**)
- Текстовое поле с маской ввода (**MaskedTextBox**)



TextBox - отображает текст и числовые поля, это наиболее часто употребляемый объект для отображения данных. Его можно создавать либо перетаскиванием из окна "**Data Sources**", либо подключить вручную. Создание этого объекта, перетаскиванием возможно почти у полей любых типов данных.

Label - полностью аналогичен объекту **TextBox**, но не позволяет изменить данные. Этот объект используется для отображения заблокированных неизменяемых полей.

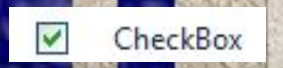


ListBox- список отображающий значения полей и позволяющий выбирать значения полей из списка. Более того, пункты списка можно задавать, используя другой источник данных.

ComboBox - объект подобный объекту **ListBox**, однако информация отображается не в списке, а выпадающем списке.

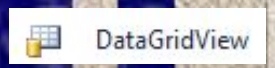
Замечание: Объекты **ListBox** и **ComboBox** могут использоваться для заполнения полей с кодами, то есть списки заполняются информацией из одной таблицы, а при выборе пункта списка его код подставляется в другую таблицу. Для этого на форму располагают не подключенный **ListBox** или **ComboBox**, затем открываем его меню действий. В меню действий в указанной последовательности выполняют следующие шаги:

- Установить галочку "**Use Data Bound Items**",
- В выпадающем списке "**DataSource**" выбрать пункт "**Other Data Source**" и там выбрать нужную таблицу.
- В выпадающем списке "**DisplayMember**" и указываем поле, которое отображается в списке.
- В выпадающем списке "**ValueMember**" указываем поле, которое подставляем при выборе пункта списка.
- В выпадающем списке "**SelectedValue**" указываем поле, куда подставляется выбранное в "**ValueMember**" значение.



CheckBox - объект используется для отображения логических полей, может быть создан перетаскиванием только для логических полей.

LinkLabel - специальный объект для отображения ссылок на адреса в Интернете. Его используют для отображения текстовых полей, если в них хранятся адреса Интернета или какой-то компьютерной сети.



DataGridView - объект, отображающий источник данных (таблицу, запрос или фильтр) в виде таблицы.

DataPicker - специальный объект, предназначенный для отображения полей типа данных "Дата/Время" в виде календаря.

MaskedTextBox - нестандартный объект, предназначенный для отображения и ввода информации по заранее заданному шаблону (маске).

Этот объект может быть создан только при помощи панели объектов и его подключение осуществляется либо перетаскиванием на него поля из окна "**Data Sources**", либо заданием его свойств вручную.

По своим свойствам он ничем не отличается от объекта **TextBox**.

Единственное дополнительное свойство у этого объекта это свойство **Mask**.

Для этого нужно щелкнуть по кнопке действий объекта в верхнем правом углу объекта. Затем в списке действий выбрать пункт "**Edit Mask**".
В появившемся окне выбрать шаблон ввода, то есть маску (**Mask**).

Замечание: Тип данных отображаемой информации должен совпадать с типом данных маски.

Программное управление информационной системой

В Visual Studio добавлять, удалять записи и перемещаться по ним можно как используя объект **Navigator**, так и используя обычные кнопки.

Пример: Создадим кнопоки для управления записями. В Visual Studio все операции с записями осуществляются с использованием объекта "**BindingSource**".

Для добавления новой записи из таблицы "Студенты" используется команда вида

```
СтудентыBindingSource.AddNew
```

Вместо метода **AddNew** можно использовать методы:

MoveNext (перейти к следующей);

MoveFirst (Перейти к первой);

MovePrevious (Перейти к предыдущей);

MoveLast (Перейти к последней);

Delete (Удалить запись).

Свойство Filter объекта BindingSource

У объекта **BindingSource** имеется свойство **Filter**. В свойстве **Filter** задаётся строка, определяющая условие отбора записей в динамических фильтрах, выполняемых на стороне клиента. Данная строка имеет следующий синтаксис:

<Поле1><Оператор1><Выражение1>

[AND|OR <Поле2><Оператор2><Выражение2>…]

Здесь:

<Поле1>, <Поле2> ... - поля на которые накладываются условия;

<Оператор1>, <Опрераторы2> - операторы сравнения, участвующие в условиях;

<Выражение1>, <Выражение2> - выражения с которыми сравниваются поля. Под *выражениями* понимаются, константы, переменные, формулы, функции и свойства объектов

Пример: Из таблицы "Студенты" необходимо отобразить студента, у которого значение поля ФИО равно "Петров".

```
СтудентыBindingSource.Filter = "ФИО = 'Петров'"
```

Свойство Filter объекта BindingSource

Обычно при формировании запроса при помощи свойства **Filter** задания условий отбора используют либо списки **ListBox**, либо выпадающие списки **ComboBox**.

Замечание: Если мы используем **ComboBox** для создания динамического фильтра, то в меню действий параметры "**Value Member**" и "**Selected Value**" настраивать не надо.

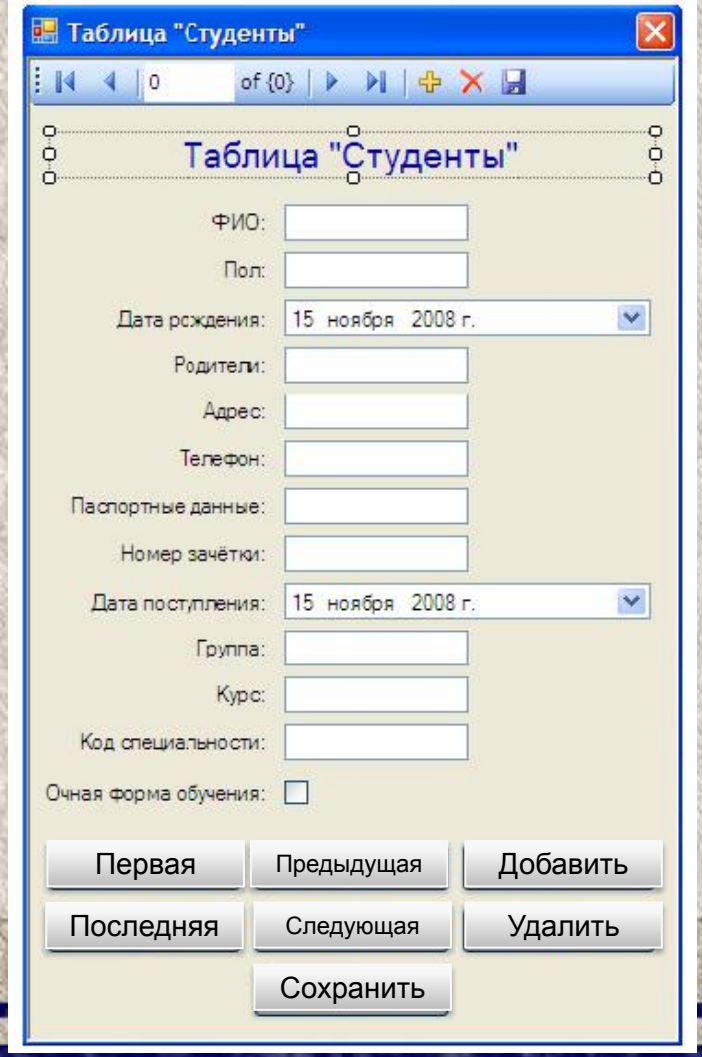
Пример: Имеется таблица "Студенты", которая отображается на форме в **DataGridView**. Необходимо на форме поместить **ComboBox** с фамилиями студентов. При выборе ФИО и нажатием на кнопку отобразить данные только по выбранному студенту.

В этом случае в меню действий **ComboBox** в параметре "**Data Source**" указываем "**Other Data Source/Студенты**". Затем в "**Display Member**" выбираем ФИО. В коде кнопки прописываем следующую команду:

```
СтудентыBindingSource.Filter = "ФИО='" &
ComboBox1.Text  & "'"
```

После нажатия кнопки в **DataGridView** отображаются данные по студенту, выбранному в выпадающем списке **ComboBox1**.

Создание сложных ленточных форм для работы с данными



C#

```
private void button1_Click(object sender, EventArgs e)
{

}
```

для перехода к первой записи

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    СтудентыBindingSource.MoveFirst()
End Sub
```

для перехода к предыдущей записи

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    СтудентыBindingSource.MovePrevious()
End Sub
```

для добавления новой записи

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    СтудентыBindingSource.AddNew()
End Sub
```

для перехода к последней записи

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    СтудентыBindingSource.MoveLast()
End Sub
```

для перехода к следующей записи

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    СтудентыBindingSource.MoveNext()
End Sub
```

для удаления текущей записи

```
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
    СтудентыBindingSource.RemoveCurrent()
End Sub
```

для сохранения изменений

```
Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click
    Me.Validate()
    Me.СтудентыBindingSource.EndEdit()
    Me.TableAdapterManager.UpdateAll(Me.StudentsDataSet)
End Sub
```

Сохранение изменений

Рассмотрим последнюю процедуру более подробно.

Она содержит следующие команды:

Me.Validate() - проверяет введенные в поля данные на соответствие типам данных полей;

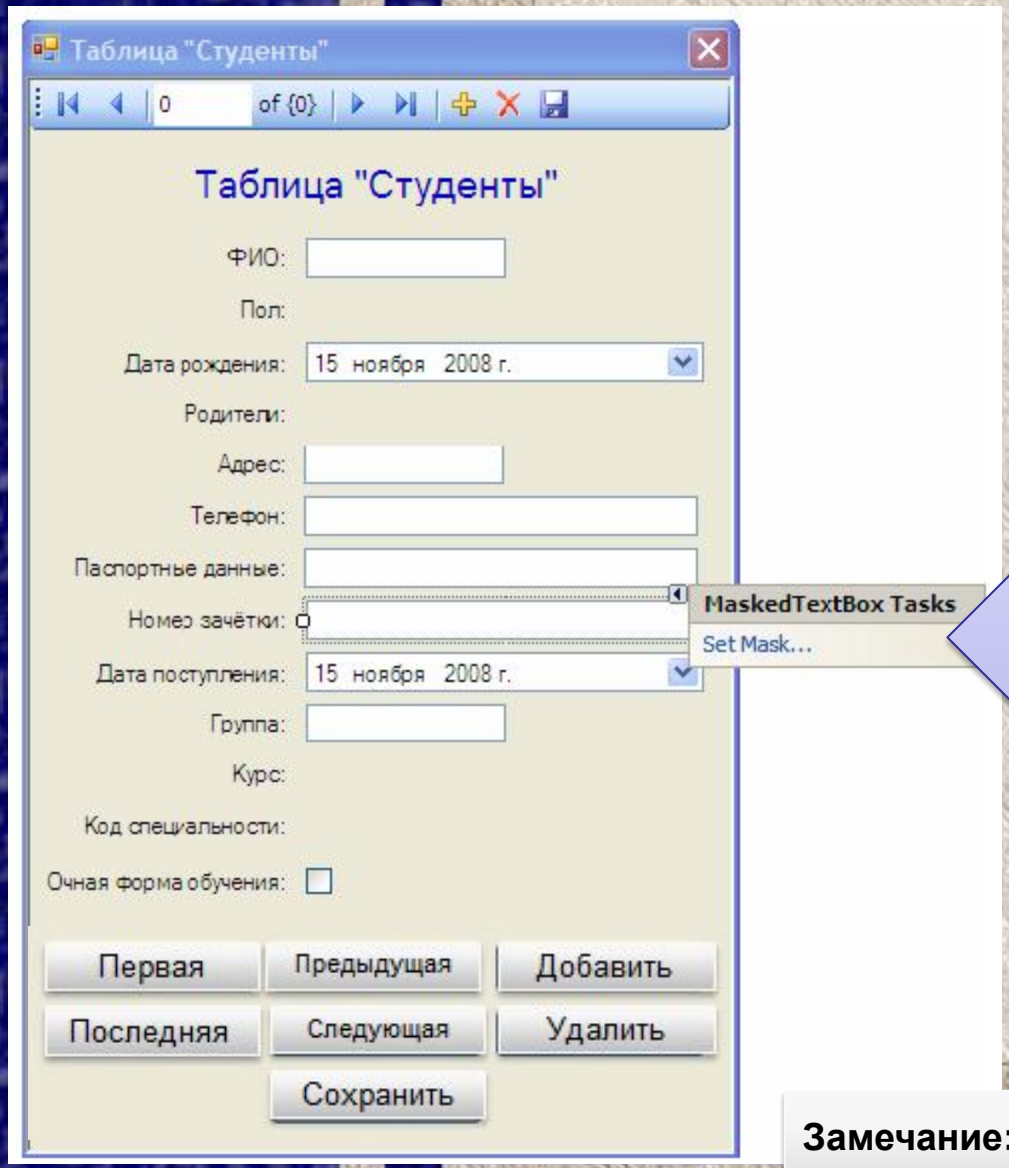
Me.СтудентыBindingSource.EndEdit() - закрывает подключение с сервером;

Me.TableAdapterManager.UpdateAll(Me.StudentsDataSet) - обновляет данные на сервере.

Использование MaskedTextBox

Для отображения полей **"Телефон"**, **"Паспортные данные"** и **"Номер зачетки"** лучше использовать текстовые поля ввода по маске (**MaskedTextBox**). Объект текстовое поле ввода по маске отсутствует в выпадающем списке объектов для отображения полей в окне **"Источники данных"**, поэтому будем создавать данные объекты при помощи панели объектов (**Toolbox**), а затем подключать их к соответствующим полям вручную.

Для создания текстовых полей ввода по маске на панели объектов используется кнопка 

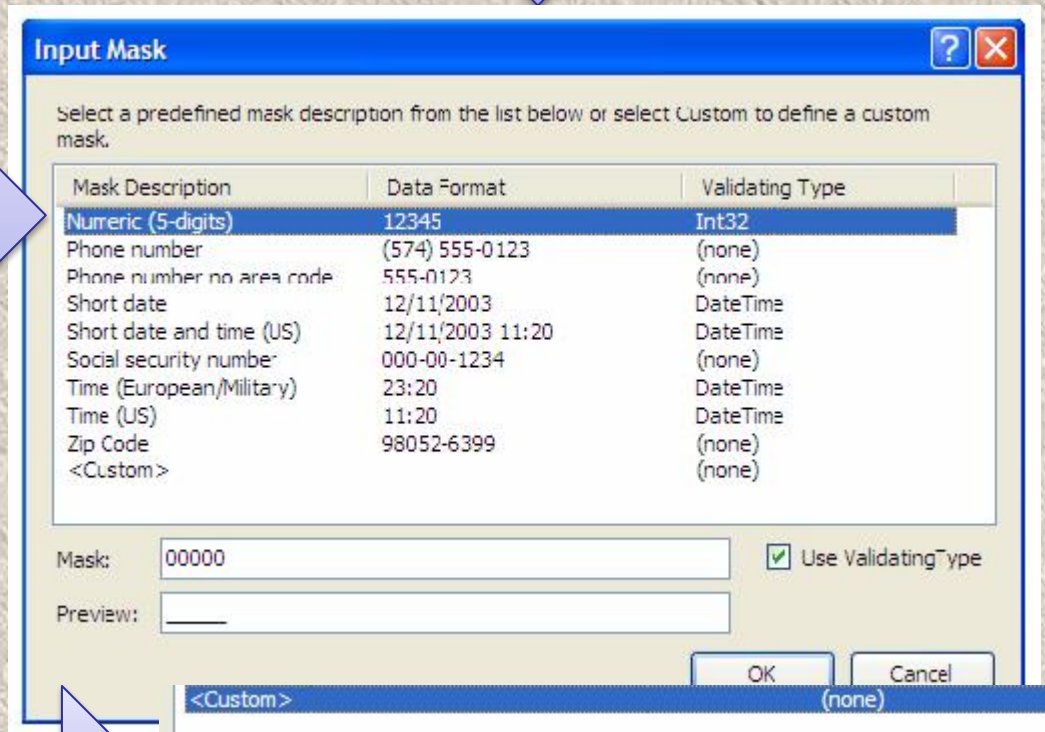


Настроим маски ввода. Начнем с объекта, отображающего номер зачетки.
На форме выделите соответствующее полю "Номер зачетки" текстовое поле ввода по маске.
Для задания маски в меню действий с объектом выберите пункт "**Set Mask**…" (Установить маску…)

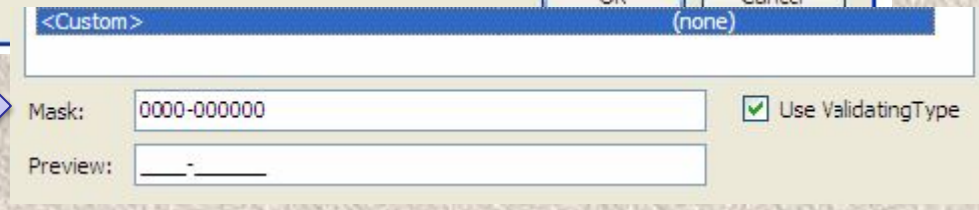
Замечание: Для отображения меню действий в верхнем правом углу объекта необходимо нажать кнопку

После выбора пункта "**Set Mask**…" на экране появится окно задания маски "**Input Mask**" (Введите маску)

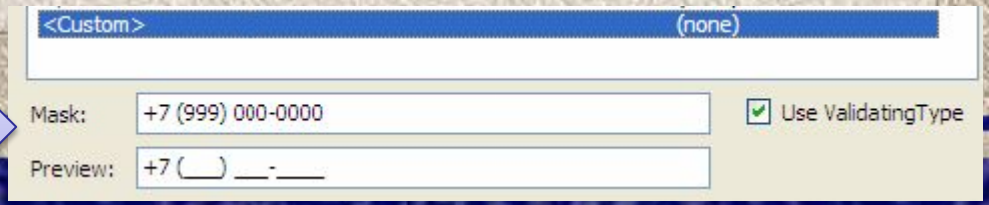
В окне "Input Mask" выберите маску "Numeric (5-digits)" (Числовое (5-цифр)) и нажмите кнопку "**Ok**"



маска для текстового поля ввода по маске отображающего поле **"Паспортные данные"**



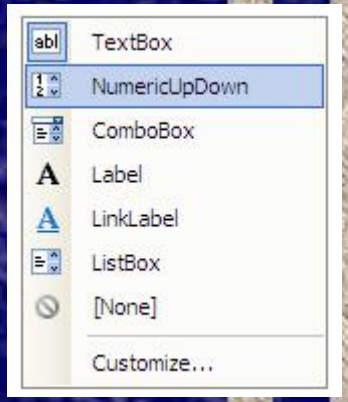
маска для текстового поля ввода по маске отображающего поле **"Телефон"**



Использование MaskedTextBox

Теперь необходимо подключить созданные текстовые поля ввода по маске к соответствующим полям.

Для этого с панели **"Источники данных" (DataSources)** перетащите поле **"Номер зачетки"** на текстовое поле ввода по маске, расположенное справа от надписи **"Номер зачетки"**. Проделайте такую же операцию с полями **"Паспортные данные"** и **"Телефон"**, перетащив их на соответствующие им текстовые поля ввода по маске.

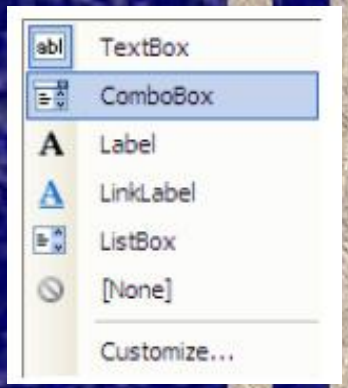


Отобразим поле **"Курс"** при помощи числового счетчика (объект **NumericUpDown**).

Для этого, на панели **"Источники данных«** необходимо нажать кнопку, расположенную справа от поля **"Курс"** и в выпадающем списке выберите объект для отображения данного поля как **"NumericUpDown".**

Затем перетащите поле на форму мышью, расположив, его справа от надписи **"Курс"**.

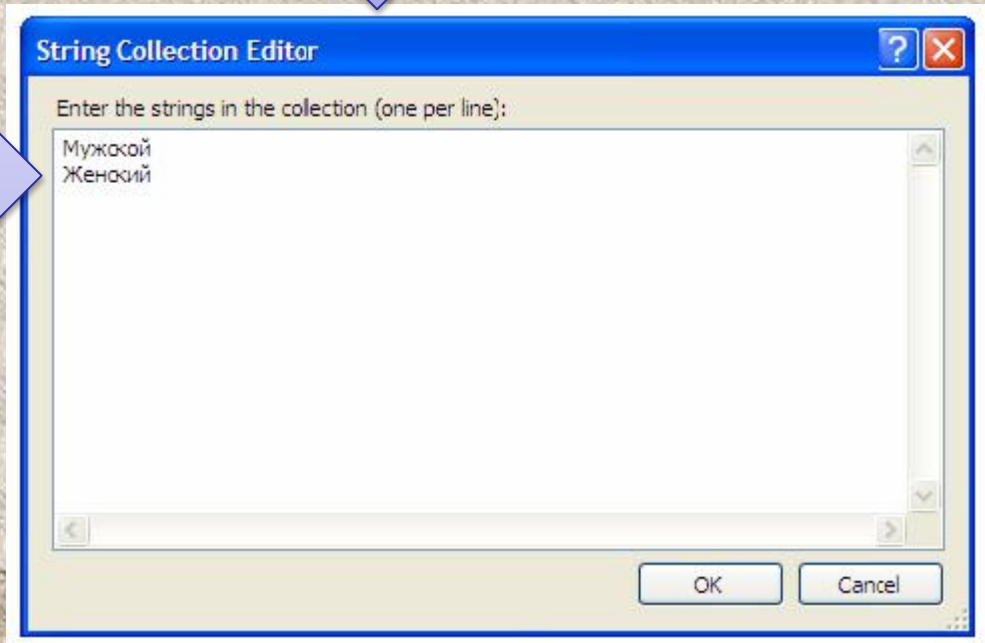
Замечание: После перетаскивания поля **"Курс"** на форму слева от него появится еще одна надпись **"Курс"**. Удалите ее, щелкнув по ней **ЛКМ**, а затем нажав кнопку **"Delete"** на клавиатуре.



Отображение полей **"Пол"** и **"Родители"** в виде выпадающих списков (Объект **ComboBox**).
Для этого, на панели **"Источники данных"** нажмите кнопку, расположенную справа от поля **"Пол"** и в выпадающем списке выберите объект для отображения данного поля как **"ComboBox".**

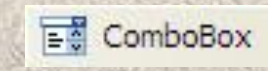
Чтобы заполнить выпадающие списки необходимо выделить выпадающий список, отображающий поле "Пол". На панели свойств (**Properties**) и нажать кнопку в свойстве "**Items**" (Элементы списка). Появится окно "**String Collection Editor**" (Редактор строковых коллекций)

В появившемся окне в отдельных строках необходимо набрать элементы выпадающего списка: "Мужской" и "Женский Затем нажмите кнопку "**Ok**".



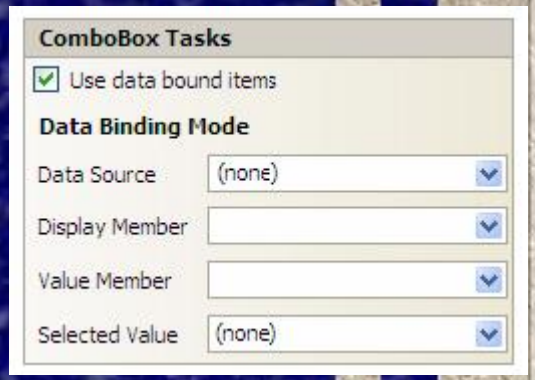
Заполнение выпадающих списков

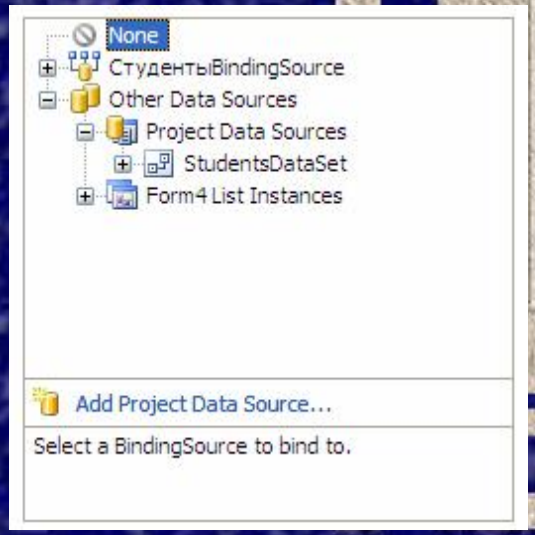
Вместо поля **"Код специальности"** отобразим специальность соответствующую заданному коду, при помощи выпадающего списка. При этом сам выпадающий список будет заполнен специальностями из таблицы **"Специальности"** и при выборе специальности ее код будет автоматически подставляться в поле **"Код специальности"** таблицы **"Студенты"**.

Поместите справа от надписи **"Код специальности"**, неподключенный ни к каким полям выпадающий список. Для создания выпадающего списка на панели объектов воспользуйтесь кнопкой 

После создание выпадающего списка подключим его к полю **"Код специальности"** из таблицы **"Студенты"** и настроим заполнение списка значениями поля **"Наименование специальности"** из таблицы студенты. Для этого выделите вновь созданный выпадающий список, отобразите меню действий и в меню действий включите опцию **"Use data bound items"** (Использовать связанные с данными элементы списка)

Заполнение выпадающих списков





В панели действий под опцией **"Use data bound items"** расположены следующие параметры:

Data Source (Источник данных) - определяет таблицу или запрос из которого заполняется список;

Display Member (Член отображения) - определяет поле значениями которого заполняется список;

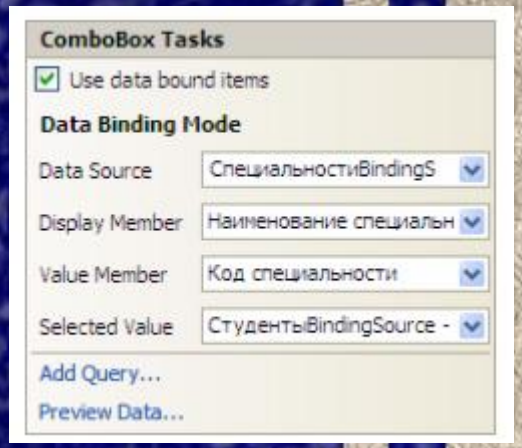
Value Member (Член значений) - определяет значения какого поля подставляются в связанное с выпадающим списком поле;

Selected Value (Выбранное значение) - определяет связанное с выпадающим списком поле.

Для изменения параметров необходимо нажать кнопку внутри поля параметра.

Появится древовидная структура выбора источника данных.

Указание параметров



В нашем случае зададим выше перечисленные параметры следующим образом:

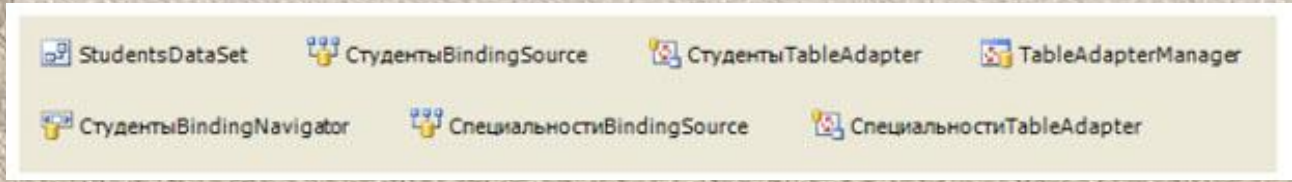
Параметр **"DataSource"** как **"Other Data Sources\Project Data Sources\StudentsDataSet\Специальности"**;

Параметр **"DataMember"** как **"Наименование специальности"**;

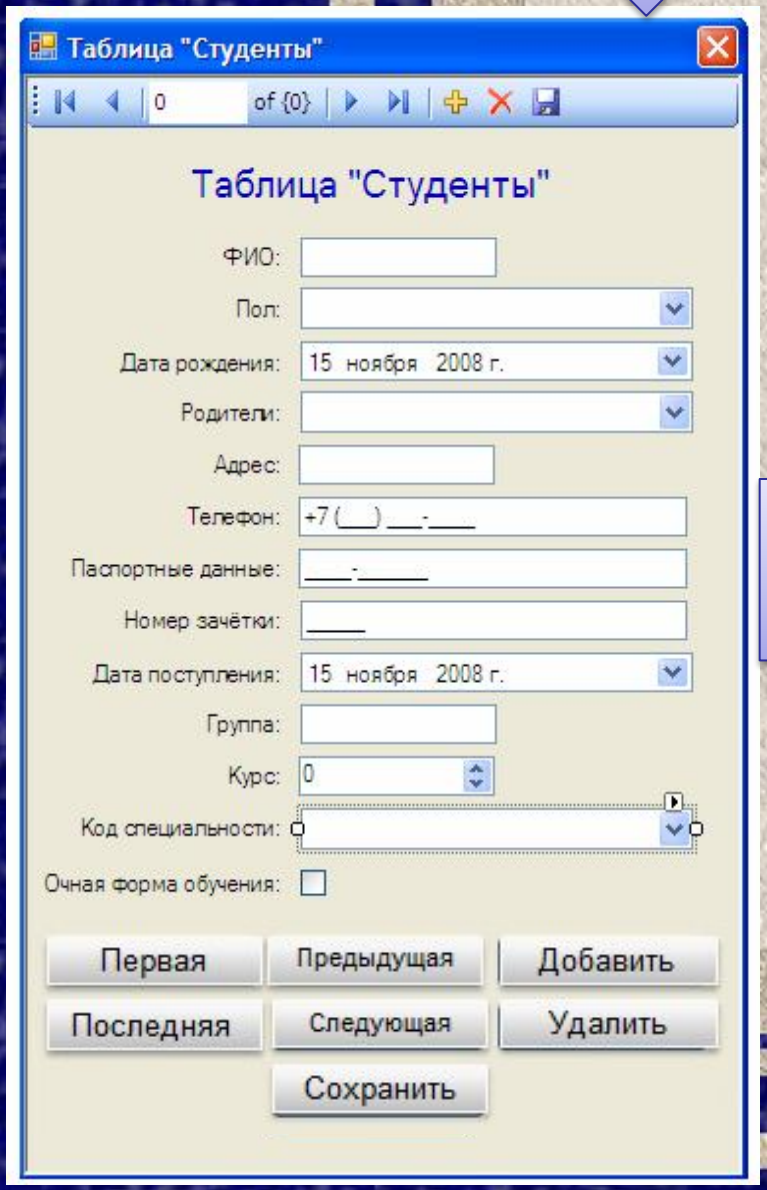
Параметр **"Value Member"** как **"Код специальности"**;

Параметр **"Selected Value"** как **"СтудентыBindingSource\Код специальности"**.

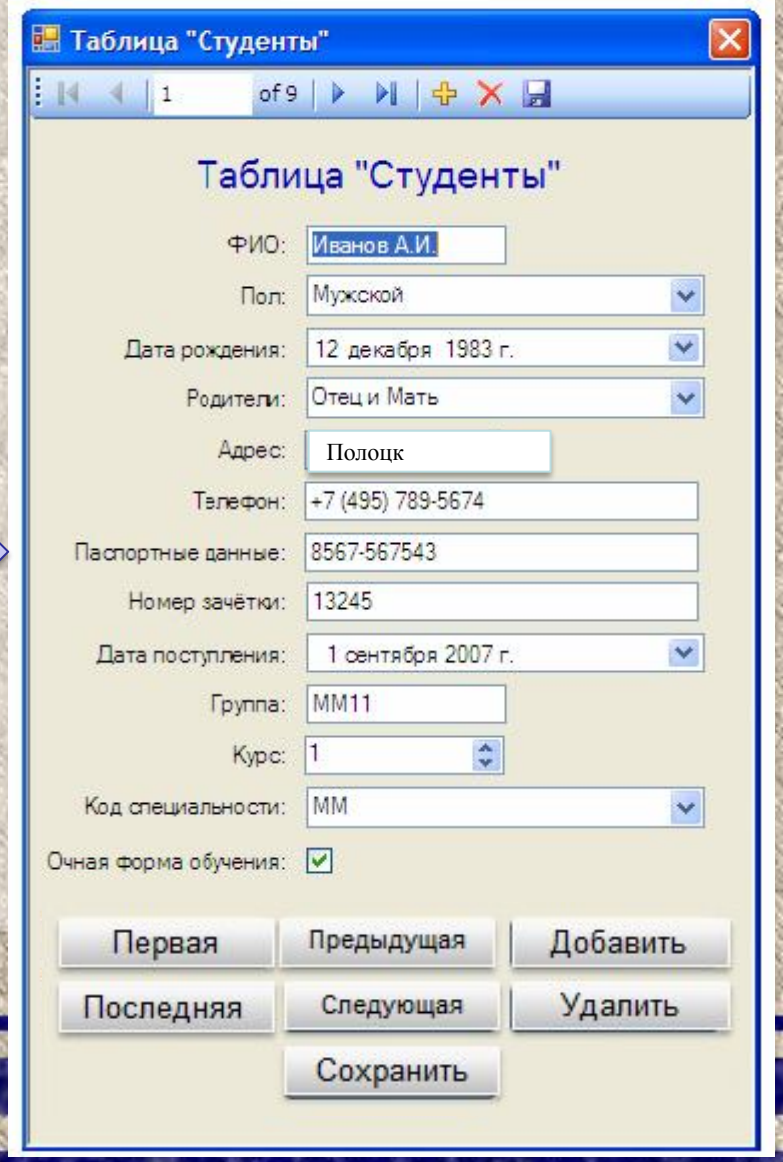
После данного действия на панели невидимых объектов, расположенной в нижней части рабочей области среды разработки, появится два новых объекта: **"СпециальностиBindingSource"** и **"СпециальностиTableAdapter"** .



После всех вышеперечисленных действий форма, отображающая таблицу "Студенты" примет вид

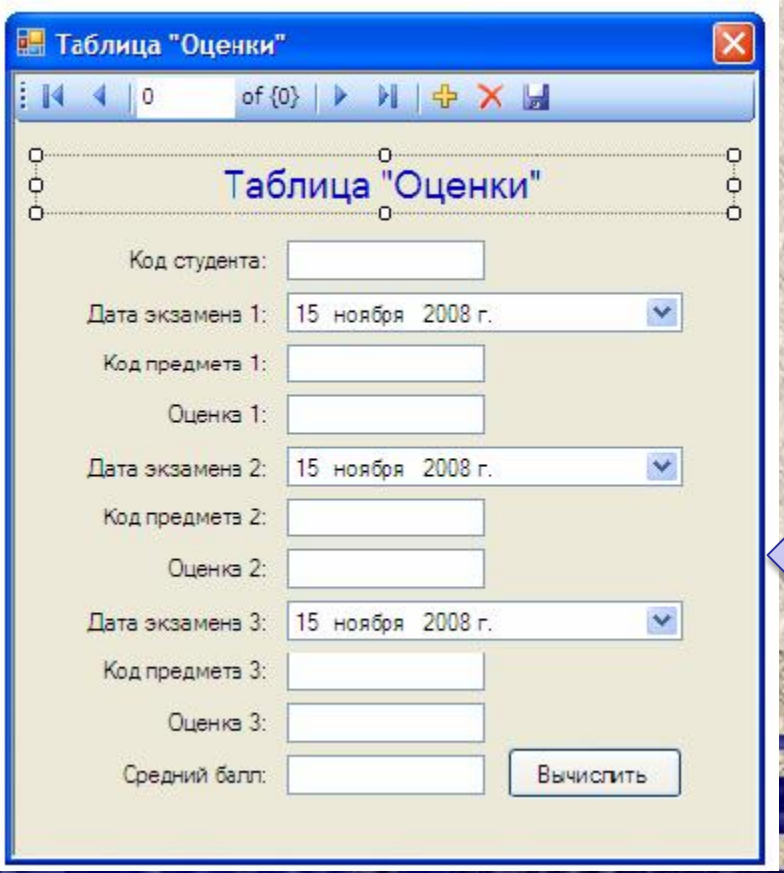


После запуска



Пример

Посмотрим реализацию вычисляемых полей. Для этого рассмотрим форму, отображающую таблицу **"Оценки".** Рассмотрим вычисление поля **"Средний балл"** на основе среднего трех полей:



Отобразим форму для таблицы "**Оценки**", щелкнув ЛКМ по ее вкладке в верхней части рабочей области среды разработки. На форму, справа от поля "**Средний балл**" поместим кнопку Button (свойство **"Text"** у вновь созданной кнопки как **"Вычислить "**).

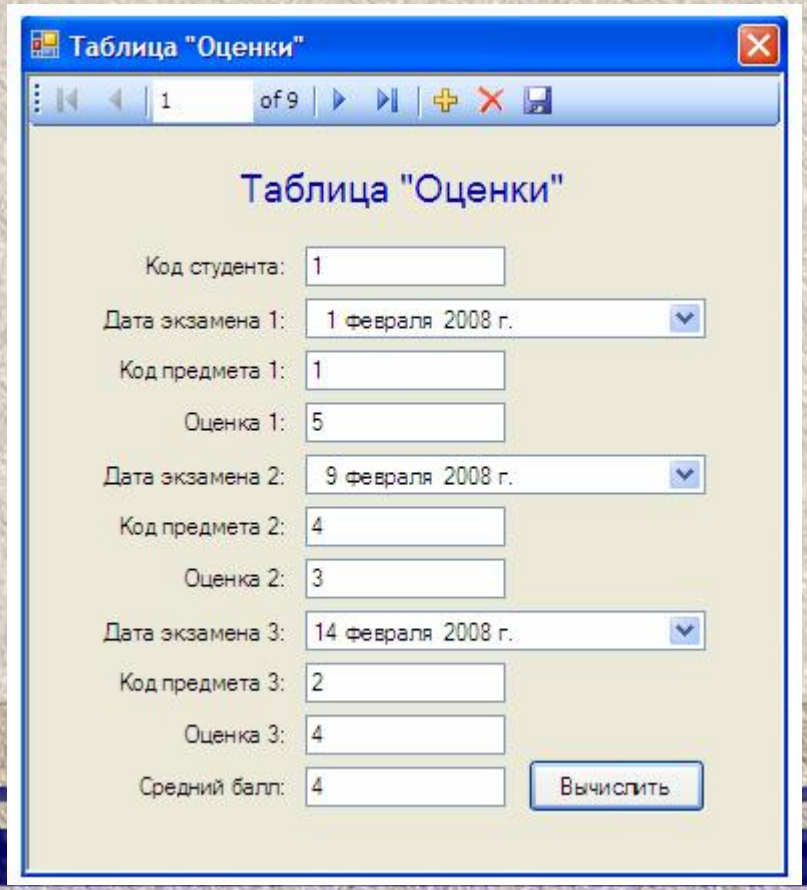
C#

```
private void button1_Click(object sender, EventArgs e)
{

}
```

Теперь дважды щелкните **ЛКМ** по кнопке **"Вычислить"** и в появившемся коде процедуры **"Button1_Click"** наберите код, вычисляющий среднее оценок.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Средний_баллTextBox.Text = (Val(Оценка_1TextBox.Text) + Val(Оценка_2TextBox.Text) + Val(Оценка_3TextBox.Text)) / 3
End Sub
```



§2.2 Создание интерфейса пользователя
Создание табличной формы

§2.2.1 Объект для отображения табличной информации DataGridView. Настройка свойств столбцов в DataGridView

Объект для отображения табличной информации DataGridView

Объект DataGridView предназначен для отображения всей информации из таблиц, запросов или фильтров на форме в виде таблицы.

Этот объект может быть создан как:

- вручную (с последующим его подключением),
- перетаскиванием всего источника данных из окна "**Data Sources**".

Однако наиболее часто его создают перетаскиванием всей таблицы, запроса или фильтра из окна "**Data Sources**" на форму.

При перетаскивании этого объекта на форму, как и в случае с другими объектами появляется панель навигации. Она выполняет функции: перемещение по записям, добавление, удаление и сохранение записей. После создания объекта **DataGridView** можно настраивать как свойства всего объекта, так и свойства отдельных столбцов.

Свойства объекта DataGridView

Настройка свойств объекта **DataGridView** осуществляется в основном через меню действий. Возможны следующие настройки:

- **Chose Data Source** - источник данных, отображаемый в таблице;
- **Enable Adding** - добавлять записи;
- **Enable Deleting** - разрешается пользователям удалять записи;
- **Enable Editing** - разрешается пользователям изменять значения полей таблицы;
- **Enable Column Reordering** - разрешается пользователям изменять порядок столбцов, просто перетаскивая их мышью.

Меню действий объекта DataGridView

Также в меню действий возможны следующее действия с таблицей:

- **Dock in parent container** - вписать объект в форму;
- **Preview Data** - появляется окно с предварительным просмотром таблицы;
- **Add Query** - добавляет SQL - запрос, который выполняется на стороне клиента;
- **Add Column** - добавление нового столбца в таблицу;
- **Edit Columns** - настройка свойств отдельных столбцов таблицы.

Настройка свойств столбцов в DataGridView

Если в меню действий выбрать пункт "**Edit Columns**", то появляется окно, где можно добавлять, удалять и редактировать столбцы. Для этого в списке столбцов левой части окна выбираем столбец, а в правой - настраиваем его свойства. Наиболее часто настраиваются следующие свойства:

- **Name** - имя столбца;
- **AutoSizeMode** - подгонка ширины столбца по его содержимому;
- **ColumnType** - определяет внешний вид ячеек столбца (какой объект для отображения информации находится в ячейках столбца);
- **DataPropertyName** - имя, отображающего в столбце поля;
- **Frozen** - фиксация столбца (столбец не передвигается при прокручивании таблицы);

Настройка свойств столбцов в DataGridView

- **HeaderText** - текст заголовка столбца;
- **Width** - ширина поля;
- **MaxInputLength** - максимально вводимая длина текста;
- **MinimumWidth** - минимальная ширина столбца;
- **ReadOnly** - блокировка столбца для редактирования данных;
- **Resizable** - разрешает менять ширину столбца;
- **SortMode** - сортировка данных в таблице по этому столбцу;
- **ToolTipText** - всплывающая подсказка для столбца;
- **Visible** - делает столбец невидимым.

Замечание: Для добавления новых столбцов в окне "**Edit Columns**" необходимо нажать кнопку **Add**, а для удаления кнопку **Remove**.

Замечание: Если необходимо настроить внешний вид всех ячеек таблицы, то для этого необходимо выделить объект **DataGridView** и на панели свойств зайти в свойство **DefaultCellStyle**. Появится окно со свойствами всех ячеек таблицы.

Замечание: В объекте **DataGridView** имеется возможность сортировки данных. Для этого используется метод **Sort**, имеющий следующий синтаксис:

```
DataGridView.Sort(<Имя столбца>, <Порядок сортировки>)
```

где `DataGridView` - это имя объекта,

`<Имя столбца>` - это имя столбца (свойство Name) по которому происходит сортировка записей в таблице, параметр `<Порядок сортировки>` определяет порядок сортировки и может принимать два значения:

`System.ComponentModel.ListSortDirection.Ascending` - сортировка по возрастанию;

`System.ComponentModel.ListSortDirection.Descending` - сортировка по убыванию.

Доступ к отдельным ячейкам таблицы

Замечание: Доступ к отдельным ячейкам таблицы можно получить через подобъект **Item**. Обращение к нему осуществляется следующим образом:

```
DataGridView.Item(i,j).<Свойство>
```

Здесь `DataGridView` - это имя объекта, `i` - горизонтальная координата ячейки, а `j` - вертикальная, `<Свойство>` - это настраиваемое свойство ячейки.

Пример: В верхнюю левую ячейку таблицы записать слово "Привет" и сделать цвет текста в ячейке красным.

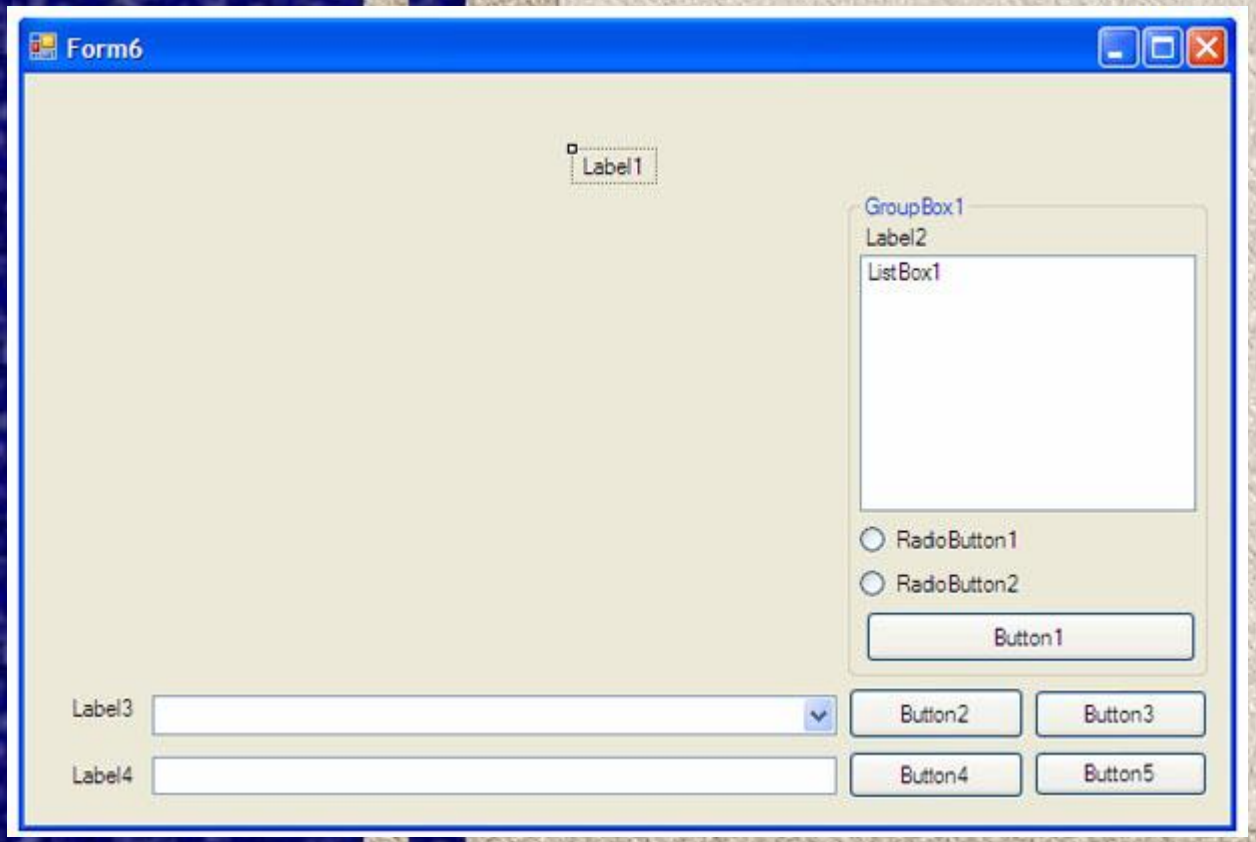
```
DataGridView.Item(0,0).Value = "Привет"
DataGridView.Item(0,0).Style.ForeColor =
Color.Red
```

Здесь `DataGridView` - это имя объекта, свойство `Value` определяет содержимое ячейки таблицы, свойство `Style.ForeColor` определяет цвет текста в ячейке. Нумерация столбцов и строк в таблице начинается с нуля.

§2.2.2 Создание табличных форм для отображения данных.

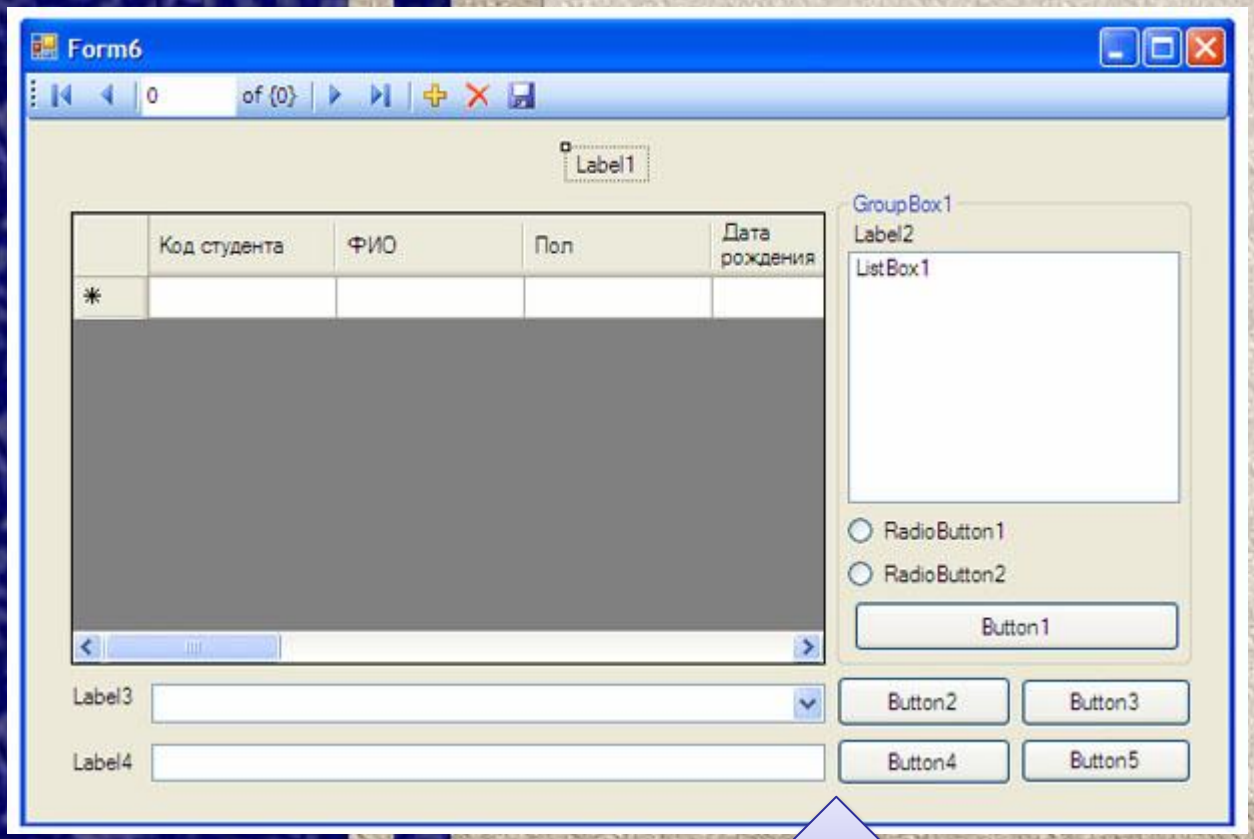
Фильтрация и сортировка данных, организация поиска информации в таблице

Рассмотрим создание табличной формы на примере формы, отображающей таблицу **"Студенты"**. Добавим в проект новую форму и на нее поместите следующие объекты:

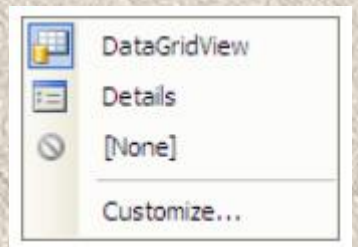


четыре надписи (**Label**),
пять кнопок (**Button**),
выпадающий список (**ComboBox**),
текстовое поле ввода (**TextBox**),
группирующую рамку (**GroupBox**),
список (**ListBox**),
два переключателя (**RadioButton**).

Добавим на форму таблицу для отображения данных (**DataGridView**) из таблицы **"Студенты"**. Для этого на панели **"Источники данных" (Data Sources)**, нажмите кнопку ▾ расположенную справа от таблицы **"Студенты"**.



В появившемся списке объектов для отображения всей таблицы выберите **"DataGridView"**



Перетащите таблицу "**Студенты**" из панели "**Источники данных**" на форму.
Форма примет следующий вид

После этих действий на форме появилась таблица для отображения данных, подключенная к таблице **"Студенты"**. Также появились объекты связи и панель навигации.

Далее настроим свойства объектов.

1. Начнем с настройки свойств формы. Задайте свойства формы следующим образом:

FormBorderStyle (Стиль границы формы): Fixed3D;

MaximizeBox (Кнопка развертывания формы во весь экран): False;

MinimizeBox (Кнопка свертывания формы на панель задач): False;

Text (Текст надписи в заголовке формы): Таблица "Студенты" (Табличный вид).

2. Зададим свойства надписей (**Label1**, **Label2**, **Label3** и **Label4**) как:

AutoSize (Авторазмер): False;

Text (Текст надписи): "Таблица "Студенты" (Табличный вид)", "Поле для сортировки", "ФИО:" и "Критерий" (Соответственно для **Label1**, **Label2**, **Label3** и **Label4**).

Для надписи **Label1** зададим:

Font (Шрифт): Microsoft Sans Serif, размер 14;

ForeColor (Цвет текста): Темно синий;

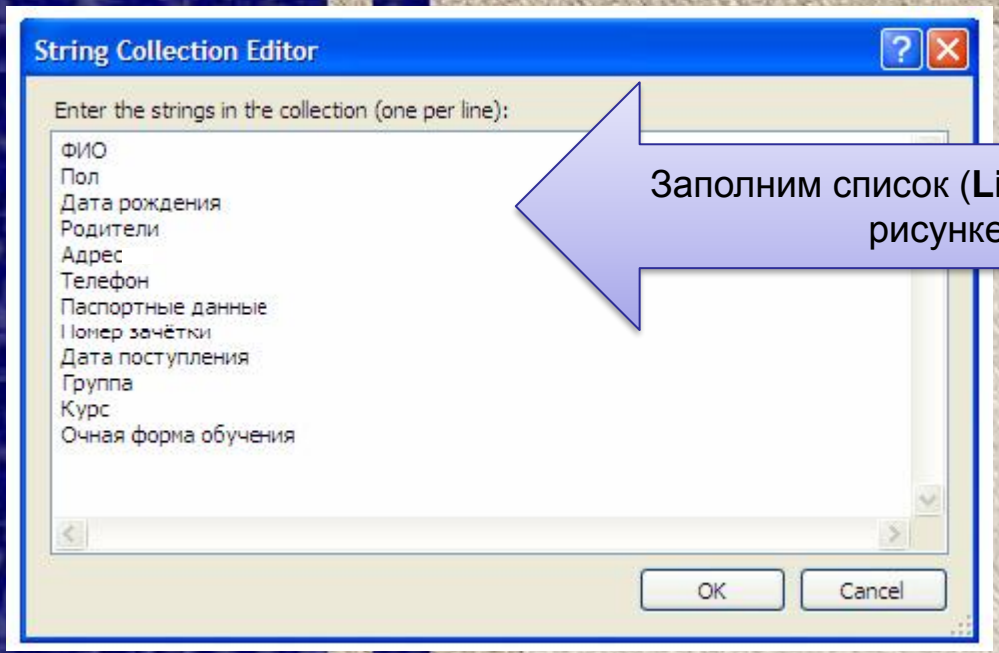
TextAlign (Выравнивание текста): MiddleCenter.

Зададим надписи на кнопках как: **"Сортировать"**, **"Фильтровать"**, **"Показать все"**, **"Найти"** и **"Закрыть"** (Соответственно для кнопок **Button1**, **Button2**, **Button3**, **Button4** и **Button5**).

Для того чтобы нельзя было произвести сортировку не выбрав поля изначально заблокируем кнопку **"Сортировать" (Button1)**.

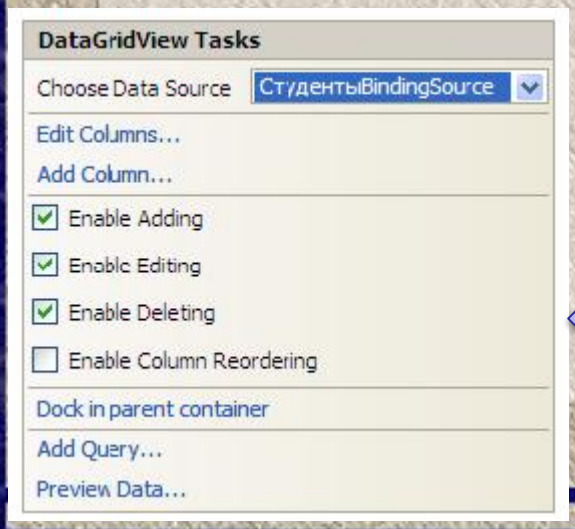
3. У группирующей рамки зададим заголовок (Свойство **Text**) равным **"Сортировка"**.

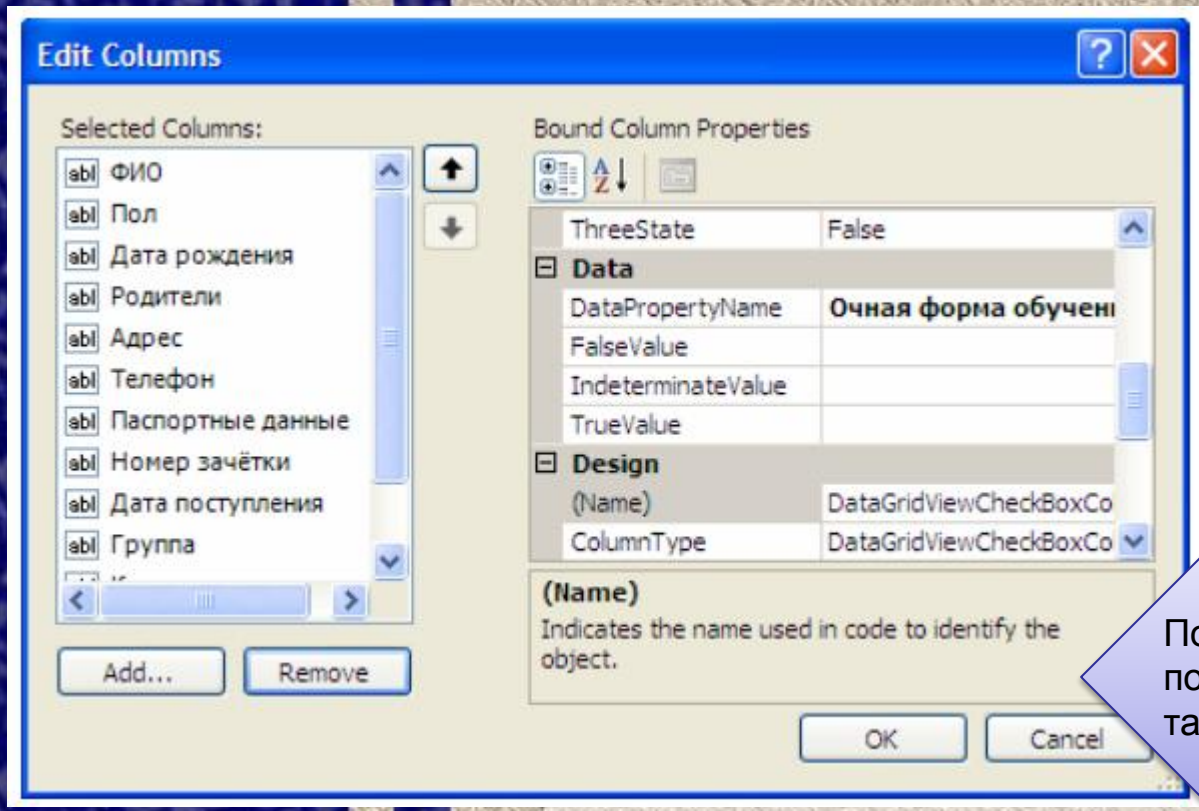
4. У переключателей (Объекты **RadioButton1** и **RadioButton2**) зададим надписи как **"Сортировка по возрастанию"** и **"Сортировка по убыванию"**, а у переключателя **"Сортировка по возрастанию" (RadioButton1)** зададим свойство **Checked (Включен)** равное **True (Истина)**.



Заполним список (**ListBox1**) значениями, представленными на рисунке, а затем нажмем кнопку "**Ok**".

Настроим таблицу для отображения данных, удалив из нее поля с кодами. Выделим таблицу на форме и отобразите ее меню действий, щелкнув **ЛКМ** по кнопке  расположенной в верхнем правом углу таблицы. В меню действий выберите пункт **"Edit columns…"**



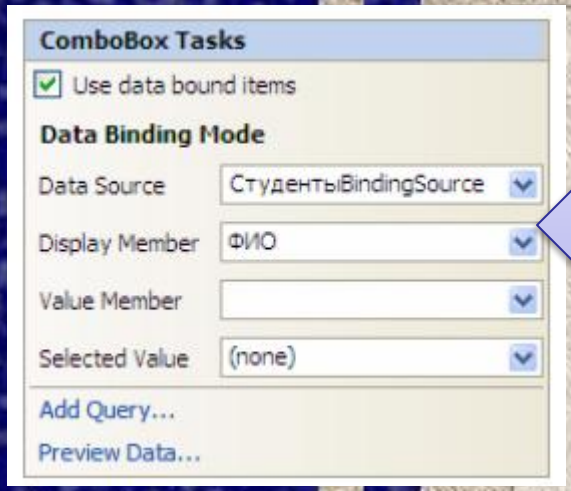


После перечисленных действий появится окно настройки свойств полей таблицы "**Edit Columns**"

В окне **"Edit Columns"** из списка полей удалите поля **"Код студента"** и **"Код специальности"**, выделив их и нажав кнопку **"Remove"** (Удалить).

Список полей примет вид показанный на рисунке.

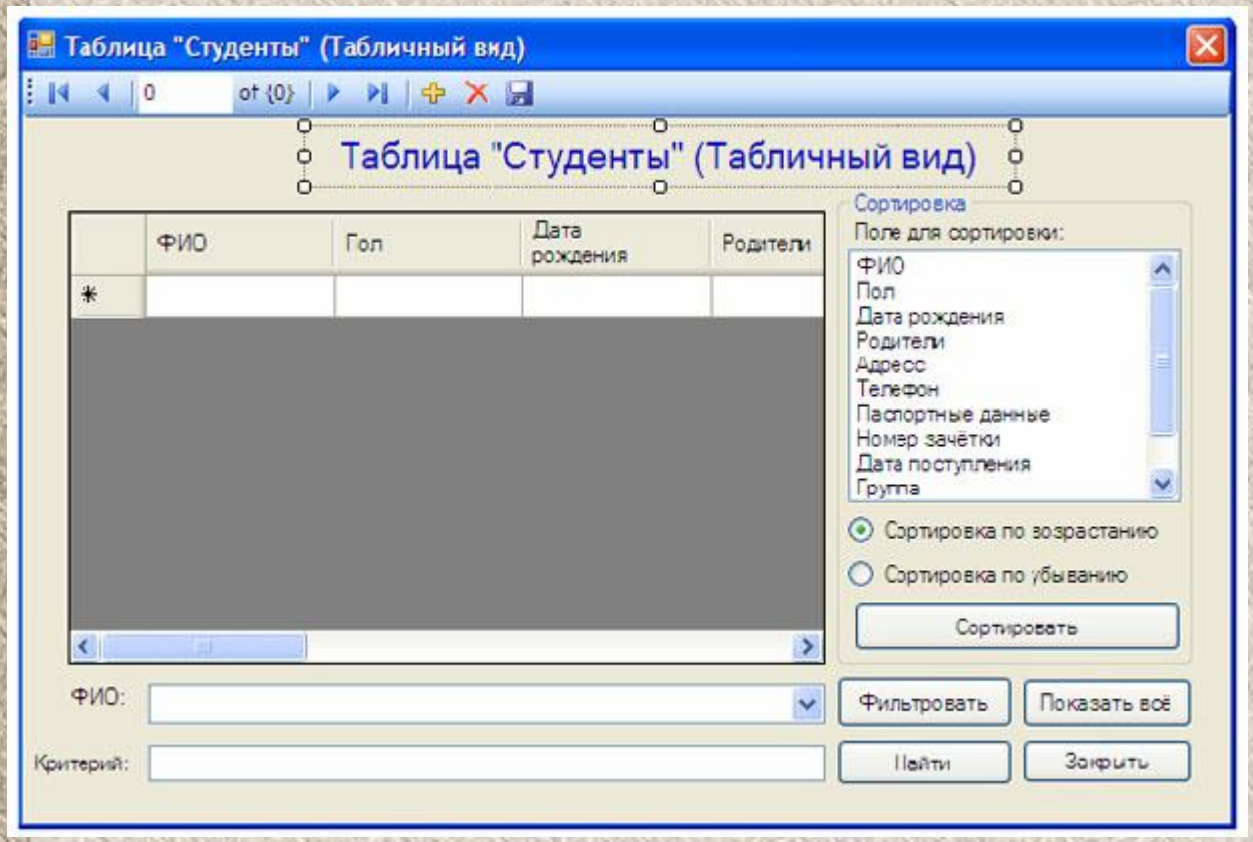
Для закрытия окна редактирования полей, и сохранения изменений нажмите кнопку "**Ok**".



Настроим заполнение выпадающего списка именами студентов из таблицы студенты. Отобразив меню действий выпадающего списка. Включим опцию "**Use Data Bound Items**". Установим параметр "**Data Source**" равным "**Other Data Sources\Project Data Sources\StudentsDataSet\Студенты**", а параметр "**Display Member**" равным "**ФИО**". Остальные параметры оставим без изменений.

Закроем окно действий выпадающего списка. На панели невидимых объектов появится дополнительный объект связи **"СтудентыBindingSource1"**, предназначенный для заполнения выпадающего списка

После настройки всех вышеперечисленных свойств объектов новая форма примет вид:



§2.2.3 Создание табличных форм для отображения данных. Написание кода обработчиков событий объектов

Разблокировка кнопки

Закончив настройку свойств объектов и перейдем к написанию кода обработчиков событий объектов.

Работу с кодом начнем с написания кода для разблокирования кнопки **"Сортировать"**, при выборе пункта списка (**ListBox1**). Для создания процедуры события дважды щелкните **ЛКМ** по списку. Появится процедура обработки события, происходящего при выборе пункта списка (**ListBox1_SelectedIndexChanged**). В процедуре наберите команду разблокировки кнопки **"Сортировать" (Button1)**:

```
Button1.Enabled = True
```

```
Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Button1.Enabled = True

End Sub
```

Сортировка данных по выбранному полю

Теперь перейдем к созданию кода сортирующего нашу таблицу в зависимости от выбранного поля и порядка сортировки при нажатии кнопки **"Сортировать"**. Дважды щелкните **ЛКМ** по кнопке **"Сортировать"**. Появится процедура **"Button1_Click"**, выполняемая при щелчке **ЛКМ** по кнопке. В процедуре наберите код:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim Col As System.Windows.Forms.DataGridViewColumn
    Select Case ListBox1.SelectedIndex
        Case 0
            Col = DataGridViewTextBoxColumn2
        Case 1
            Col = DataGridViewTextBoxColumn3
        Case 2
            Col = DataGridViewTextBoxColumn4
        Case 3
            Col = DataGridViewTextBoxColumn5
        Case 4
            Col = DataGridViewTextBoxColumn6
        Case 5
            Col = DataGridViewTextBoxColumn7
        Case 6
            Col = DataGridViewTextBoxColumn8
        Case 7
            Col = DataGridViewTextBoxColumn9
        Case 8
            Col = DataGridViewTextBoxColumn10
        Case 9
            Col = DataGridViewTextBoxColumn11
        Case 10
            Col = DataGridViewTextBoxColumn12
    End Select
    If RadioButton1.Checked Then
        СтудентыDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending)
    Else
        СтудентыDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending)
    End If

End Sub
```

Рассмотрим код более подробно:

`Dim Col As System.Windows.Forms.DataGridViewColumn` создает переменную `Col` для хранения имени выбранного столбца таблицы.

Затем следует блок `Select Case…End Select`, присваивающий в переменную `Col` имя выбранного столбца таблицы в зависимости от номера выбранного пункта списка (`ListBox1.SelectedIndex`). Если выбран первый пункт списка, то в переменную `Col` записывается столбец `DataGridViewTextBoxColumn2`, если второй, то - `DataGridViewTextBoxColumn3` и так далее.

Хотелось бы отметить тот факт, что нумерация пунктов списка начинается с нуля, а нумерация столбцов с единицы. Первый столбец **"ФИО"** носит имя `DataGridViewTextBoxColumn2`, так как имя `DataGridViewTextBoxColumn1` имеет столбец заголовков строк;

Блок `If…End If` выполняет следующую операцию: если включен переключатель **"Сортировка по возрастанию" (RadioButton1)**, то отсортировать таблицу по полю заданному в переменной `Col` по возрастанию (`СтудентыDataGridView.Sort (Col, System.ComponentModel.ListSortDirection. Ascending)`), иначе по убыванию (`СтудентыDataGridView.Sort (Col, System. ComponentModel.ListSortDirection. Descending)`).

Обработчика события нажатия кнопки "Фильтровать"

Дважды щелкните по кнопке **"Фильтровать"** и в процедуре обработки события **"Button2_Click"** наберите код:

```
СтудентыBindingSource.Filter = "ФИО='" &
ComboBox1.Text & "'"
```

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    СтудентыBindingSource.Filter = "ФИО='" & ComboBox1.Text & "'"

End Sub
```

Замечание: У объекта **СтудентыBindingSource** имеется текстовое свойство **Filter** которое определяет условие фильтрации. Условие фильтрации имеет синтаксис:

```
"<Имя поля><Оператор>'<Значение>'".
```

В нашем случае значение поля **"ФИО"** приравнивается к значению, выбранному в выпадающем списке (**ComboBox1.Text**)

Отмена фильтрации записей

Теперь перейдем к кнопке **"Показать все"**, отменяющей фильтрацию записей. Дважды щелкните по вышеперечисленной кнопке. Появится процедура **Button2_Click**. В появившейся процедуре наберите команду `СтудентыBindingSource.Filter = ""`

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    СтудентыBindingSource.Filter = ""

End Sub
```

Заметим, что если присвоить свойству **"Filter"** значение пустой строки (**""**), то его действие будет отменено.

Реализация поиска информации в таблице

Дважды щелкните по кнопке "**Найти**". В появившейся процедуре обработки нажатия кнопки "**Button4_Click**" наберите следующий код:

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    For i = 0 To СтудентыDataGridView.ColumnCount - 1
        For j = 0 To СтудентыDataGridView.RowCount - 1
            СтудентыDataGridView.Item(i, j).Style.BackColor = Color.White
            СтудентыDataGridView.Item(i, j).Style.ForeColor = Color.Black
        Next j
    Next i
    For i = 0 To СтудентыDataGridView.ColumnCount - 1
        For j = 0 To СтудентыDataGridView.RowCount - 1
            If InStr(СтудентыDataGridView.Item(i, j).Value, TextBox1.Text) Then
                СтудентыDataGridView.Item(i, j).Style.BackColor = Color.AliceBlue
                СтудентыDataGridView.Item(i, j).Style.ForeColor = Color.Blue
            End If
        Next j
    Next i

End Sub
```

Данная процедура состоит из двух частей:

Первый блок **`For i=0…..Next i`**. перебирает все ячейки таблицы и устанавливает в них белый цвет фона и черный цвет текста. То есть, отменяет результаты предыдущего поиска;

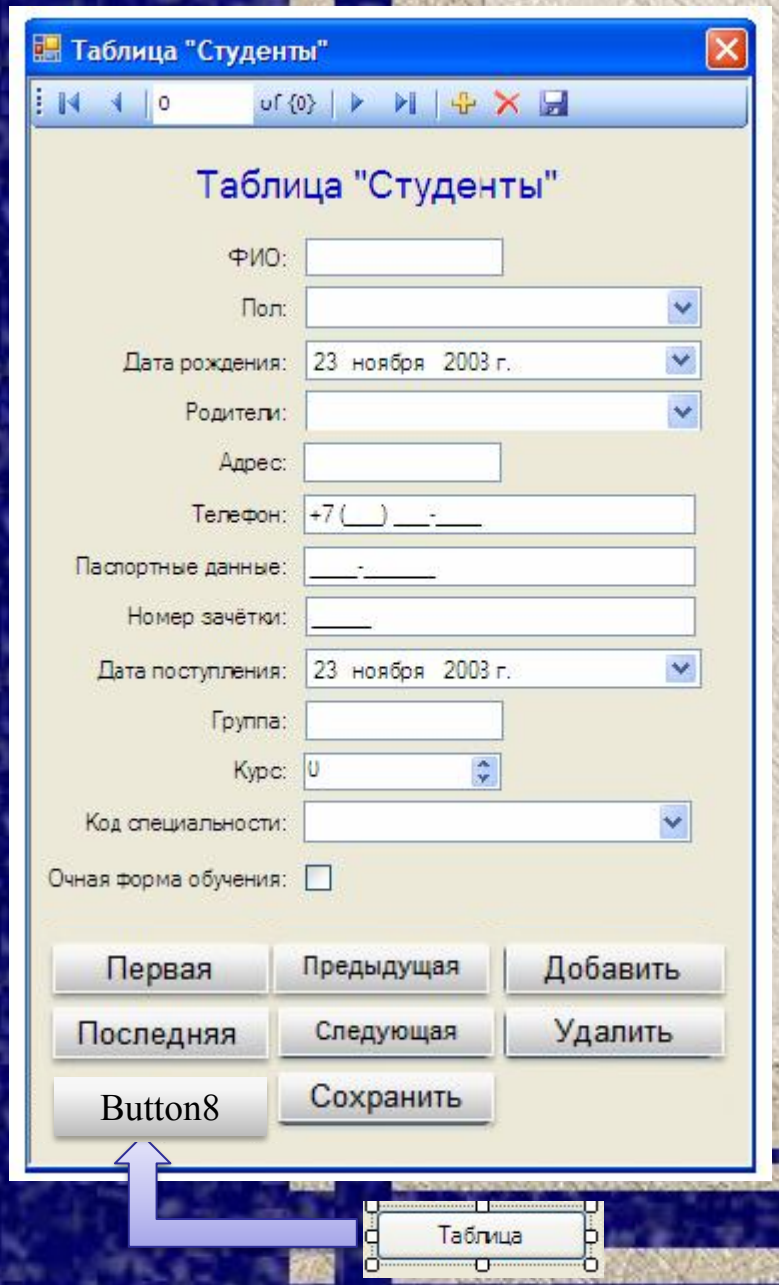
Второй блок **`For i=0…..Next i`**. перебирает все ячейки таблицы и если они содержат текст, введенный в поле ввода (**TextBox1**), то устанавливает в них голубой цвет фона и синий цвет текста, чем выделяет искомые ячейки.

Кнопка "Закрыть"

Наконец рассмотрим код для кнопки **"Закрыть"**. Дважды щелкните **ЛКМ** по этой кнопке и в появившейся процедуре **"Button5_Click"** наберите команду "`Me.Close()`", закрывающую выше рассматриваемую форму

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    Me.Close()

End Sub
```

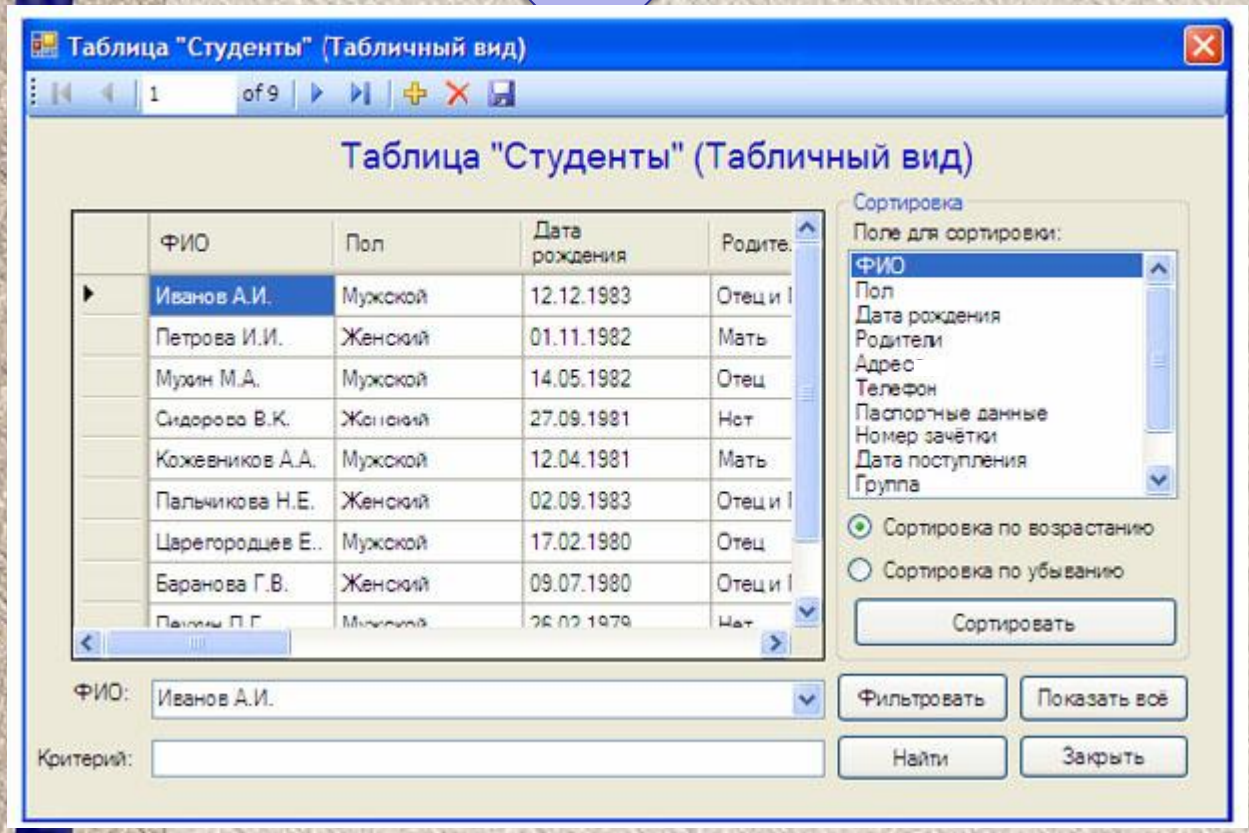


В заключение создадим кнопку на ленточной форме, отображающей таблицу **"Студенты"**, для отображения соответствующей табличной формы. Откройте ленточную форму для таблицы **"Студенты" (Form4)** и поместите на нее новую кнопку, как это показано на рисунке. Задайте надпись у новой кнопки (свойство **Text**), как **"Таблица"**.

Подключим к кнопке **"Таблица"** созданную ранее табличную форму (**Form6**). Для этого дважды щелкните **ЛКМ** по кнопке **"Таблица"** и в появившейся процедуре **"Button8_Click"** наберем команду "**`Form6.Show`**".

```
Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Form6.Show()
End Sub
```

Проверим работоспособность созданной табличной формы. Запустим проект и на главной кнопочной форме нажмем кнопку "**Таблица "Студенты""**. На появившейся ленточной форме, отображающей таблицу "**Студенты**" нажмем кнопку "**Таблица**".
Появится новая табличная форма



Отметим тот факт, что после проведения всех вышеописанных действий панель обозревателя проекта (**Solution Explorer**) примет вид