**Лабораторная работа №5. Работа с объектом DataSet.**

**Цель работы:**

– получить практические навыки создания и использования таблиц в объекте DataSet.

**Теоретические сведения**

1. **Объект DataSet**

Объект DataSet  предназначен для хранения данных, считанных с помощью DtaAdapter. DataSet содержит в себе два важных ингредиента: коллекцию из нуля или более таблиц (доступных через свойство Tables) и коллекцию из нуля или более отношений, которые можно применять для связывания таблиц между собой (представленных свойством Relationships).

Каждая запись коллекции DataSet.Tables — это DataTable. Объект DataTable содержит собственные коллекции — коллекцию Columns объектов DataColumn (описывающие имя и тип данных каждого поля) и коллекцию Rows объектов DataRow(содержащих действительные данные каждой записи).

Каждая Запись в DataTable представлена объектом DataRow. Каждый объект DataRow представляет одиночную запись в таблице, которая была извлечена из источника данных. DataRow является контейнером для действительных значений полей. Получить доступ к ним можно по имени поля, как в случае myRow["FieldName"]. DataSetявляется автономным объектом не связанным с источником объектом. Все изменения проводятся локально в объекте DataSet, расположенном в памяти. DataSet никогда не полагается на какое-либо соединение с источником данных.

Объект DataSet также имеет методы, которые могут писать и читать данные и схемы XML, а также методы для быстрой очистки и дублирования данных. Эти методы кратко описаны в таблице ниже:

|  |  |
| --- | --- |
| Некоторые методы класса DataSet | |
| Метод | Описание |
| Clear() | Очищает все данные таблиц. Однако этот метод оставляет нетронутой информацию о схеме и отношениях |
| Copy() | Возвращает точный дубликат DataSet с тем же набором таблиц, отношений и данных |
| Clone() | Возвращает DataSet с той же структурой (таблицами и отношениями), но без данных |
| Merge() | Принимает другой DataSet, DataTable или коллекцию объектов DataRow и объединяет с текущим объектом DataSet, добавляя новые таблицы и объединяя данные в существующих |

1. **Наполнение объекта DataSet**

В следующем примере показано, как данные из таблицы SQL Server можно использовать для наполнения объекта DataTable, принадлежащего DataSet.

Предполагается, что создана форма WindowsForms, в которой имеется список (ListBox) с именем list1 и кнопка чтения данных с именем btFill.Обработчик события нажатия кнопки чтения имеет вид:

***private void btFill\_Click(object sender, Event Args e)***

***{***

***// Организация подключения***

***string connectionString = "Data Source=MyPC;Initial Catalog=trainingDB;Integrated Security=True";***

***SqlConnection con = new SqlConnection(connectionString);***

***string command = "SELECT \* FROM Employees";***

***SqlDataAdapter adapter = new SqlDataAdapter(command, con);***

***// Заполнение tDataSet***

***DataSet dataset = new DataSet();***

***adapter.Fill(dataset, "Employees");***

***// Вывод из DataSet строк таблицы Employees в элемент list1***

***foreach (DataRow row in dataset.Tables["Employees"].Rows)***

***{***

***string result = row["id"].ToString() + " " + row["LName"] + " " + row["FName"]+" "+ row["MName"];***

***list1.Items.Add(result); }***

***}***

Сначала код создает соединение и определяет текст запроса SQL. На следующем шаге создается новый экземпляр класса SqlDataAdapter, который извлечет список сотрудников. После этого создается новый пустой объект DataSet и с помощью метода DataAdapter.Fill() выполняется запрос с помещением результата в новый объект DataTable этого DataSet. В этот момент можно также указать имя таблицы. Если этого не сделать, автоматически будет использовано имя по умолчанию (наподобие Table). В примере имя таблицы соответствует имени исходной таблицы базы данных.

В коде не открывается соединение за счет вызова Connection.Open(). Вместо этого DataAdapter открывает и закрывает связанное с ним соединение автоматически, когда вызывается метод Fill().

Последний шаг — отображение содержимого DataSet.

Внешний вид формы с отображением прочитанных данных показан на рисунке 1.

Форма показана на рис.6.1

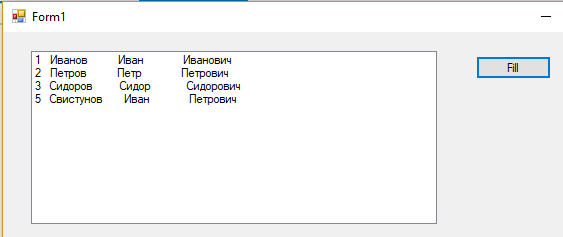


Рисунок 1. Внешний вид формы, отображающей данные из DataSet

1. **Работа с объектом DataSet**

C объектами DataSet и DataTable можно работать без какой-либо базы данных. В примере показано создание таблицы в DataSet вручную/

***private void btTables\_Click(object sender, Event Args e)***

***{***

***DataSet bookStore = new DataSet("BookStore");***

***DataTable booksTable = new DataTable("Books");***

***// добавление таблицы в dataset***

***bookStore.Tables.Add(booksTable);***

***// создание столбцов для таблицы Books***

***DataColumn idColumn = new DataColumn("Id", Type.GetType("System.Int32"));***

***idColumn.Unique = true; // столбец будет иметь уникальное значение***

***idColumn.AllowDBNull = false; // не может принимать null***

***idColumn.AutoIncrement = true; // будет автоинкрементироваться***

***idColumn.AutoIncrementSeed = 1; // начальное значение***

***idColumn.AutoIncrementStep = 1; // приращении при добавлении новой строки***

***DataColumnname Column = new DataColumn("Name", Type.GetType("System.String"));***

***DataColumn priceColumn = new DataColumn("Price", Type.GetType("System.Decimal"));***

***priceColumn.DefaultValue = 100; // значение по умолчанию***

***DataColumn discountColumn = new DataColumn("Discount", Type.GetType("System.Decimal"));***

***discountColumn.Expression = "Price \* 0.2";***

***booksTable.Columns.Add(idColumn);***

***booksTable.Columns.Add(nameColumn);***

***booksTable.Columns.Add(priceColumn);***

***booksTable.Columns.Add(discountColumn);***

***// определение первичного ключа таблицы books***

***booksTable.PrimaryKey = new DataColumn[] { booksTable.Columns["Id"] };***

***DataRow row = booksTable.NewRow();***

***row.ItemArray = new object[] { null, "Война и мир", 200 };***

***booksTable.Rows.Add(row); // добавление первой строки***

***booksTable.Rows.Add(new object[] { null, "Отцы и дети", 170 }); // добавление второй строки***

***booksTable.Rows.Add(new object[] { null, "Преступление и наказание",270 }); // добавление //третьей строки***

***booksTable.Rows.Add(new object[] { null, "Идиот",270 }); // добавление четвертой строки***

***booksTable.Rows.Add(new object[] { null, "Арзипелаг ГУЛАГ", 3000 }); // добавление пятой //строки***

***foreach (DataRow r in booksTable.Rows)***

***{***

***string result = "";***

***foreach (var cell in r.ItemArray)***

***result += cell + " ";***

***list2.Items.Add(result);***

***}***

***}***

В приведенном коде сначала создаются объекты DataSet и DataTable, в конструктор которых передается название. Затем создается четыре столбца. Каждый столбец в конструкторе принимает два параметра: имя столбца и его тип.

|  |  |
| --- | --- |
|  | ***DataColumnidColumn = new DataColumn("Id", Type.GetType("System.Int32"));*** |

Причем для столбца Id устанавливается, что значения этого столбца должны иметь уникальное значение, не должны принимать null, и их значение при добавлении нового объекта будет инкрементироваться на единицу. То есть фактически это стандартный столбец Id, как в большинстве баз данных.

Далее создается еще три столбца, при этом для столбца Discount устанавливается свойство Expression, указывающее на выражение, которое будет использоваться для вычисления значения столбца:

|  |  |
| --- | --- |
|  | ***discountColumn.Expression = "Price \* 0.2";*** |

То есть в данном случае значение столбца Discount равно значению столбца Price, умноженного на 0.2.

Затем устанавливается первичный ключ для таблицы с помощью свойства PrimaryKey:

|  |  |
| --- | --- |
|  | ***booksTable.PrimaryKey = new DataColumn[] { booksTable.Columns["Id"] };*** |

В роли первичного ключа выступает столбец Id. После определения схемы таблицы в нее добавляются две строки:

|  |  |
| --- | --- |
|  | ***DataRow row = booksTable.NewRow();***  ***row.ItemArray = new object[] { null, "Войнаимир", 200 };***  ***booksTable.Rows.Add(row); // добавляемпервуюстроку***  ***booksTable.Rows.Add(new object[] {null, "Отцыидети", 170 });*** |

Значения в метод booksTable.Rows.Add можно передать как напрямую в виде массива объектов, так и в виде объекта DataRow. При этом нам надо передать ровно столько значений, сколько в таблице столбцов. Однако поскольку первый столбец Id устанавливается через автоинкремент, можно передать значение null - оно все равно будет игнорироваться. Также можно опустить последний параметр для столбца Discount, так как его значение вычисляется с помощью выражения "Price \* 0.2". Более того можно не указывать и значение для третьего столбца Price, так как у него установлено свойство DefaultValue, которое устанавливает значение по умолчанию, если значение отсутствует:

|  |  |
| --- | --- |
|  | ***booksTable.Rows.Add(new object[] {null, "Отцы и дети"});*** |

И в конце идет перебор строк таблицы.

Кроме добавления мы можем производить и другие операции со строками. Например, мы можем получить строку по индексу:

|  |  |
| --- | --- |
|  | ***DataRow row = booksTable.Rows[0]; // первая строка*** |

Получив строку по индексу, можно изменить ее ячейки:

|  |  |
| --- | --- |
|  | ***booksTable.Rows[0][2] = 300;*** //третьей ячейке первой строки присваивается значение 300 |

И также можно удалять строку:

|  |  |
| --- | --- |
|  | ***booksTable.Rows.RemoveAt(1); // удаление второй строки по индексу***  ***// другой сопосб удаления***  ***DataRow row = booksTable.Rows[0];***  ***booksTable.Rows.Remove(row);*** |

Внешний вид формы показан на рис. 2.

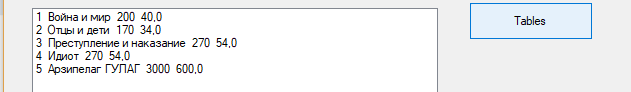


Рисунок 2. Внешний вид формы, отображающий данные из созданной в DataSet таблицы

**4. Поиск определенных строк**

Класс DataTable предлагает удобный метод Select(), позволяющий искать строки на основе SQL-выражения. Выражение, которое используется с методом Select(), играет ту же роль, что и конструкция WHERE в операторе SELECT, но имеет дело с данными в памяти, находящимися в объекте DataTable (так что никаких операций с базой данных не производится).

Например, следующий код извлекает все книги, которые продаются со скидкой более 50:

***public DataSet bookStore = new DataSet("BookStore");***

***private void btDiscount\_Click(object sender, EventArgs e)***

***{***

***// Получить книги сос скидкой более 50***

***DataRow[] rows = bookStore.Tables["Books"].Select("Discount > 50");***

***// Отобразить***

***foreach (DataRow r in rows)***

***{***

***string result = "";***

***foreach (var cell in r.ItemArray)***

***result += cell + " ";***

***list3.Items.Add(result);***

***}***

***}***

В этом примере в методе Select() используется достаточно простая строка фильтра. Для того, чтобы использовать один и тот же объект DataSet в разных функциях, его необходимо описать вне тела функции с модификатором public.

Метод Select() имеет одно потенциальное ограничение — он не принимает параметризованных условий. Если фильтрующее выражение содержит значение, передаваемое пользователем, итерацию по DataTable для нахождения требуемых строк нужно выполнять вручную, не пользуясь методом Select().

1. **Обновление источника данных**

Получив данные в DataSet, мы можем производить с ними различными операции: удалять, изменять, добавлять новые записи. Однако все делаемые нами изменения автоматически не будут сохраняться в БД. Для этого нам еще надо вызвать метод **Update** объекта SqlDataAdapter, который заполнял DataSet.

Для модификации данных в БД в соответствии с изменениями в DataSet SqlDataAdapter использует команды InsertCommand, UpdateCommand и DeleteCommand. Можно вручную определить для этих команд sql-выражения, либо воспользоваться классом **SqlCommandBuilder**, который позволяет автоматически сгенерировать нужные выражения. Пример использования SqlCommandBuilder:

***public Form2()***

***{***

***InitializeComponent();***

***}***

***static public string connectionString = "Data Source=MyPC;Initial Catalog=trainingDB;Integrated Security=True";***

***static SqlConnection con = new SqlConnection(connectionString);***

***static string command = "SELECT \* FROM Employees";***

***static SqlDataAdapter adapter = new SqlDataAdapter(command, con);***

***public DataSet ds = new DataSet();***

***private void btRead\_Click(object sender, EventArgs e)***

***{***

***// Заполнение DataSet***

***// DataSet dataset = new DataSet();***

***adapter.Fill(ds, "Employees");***

***// Вывод из DataSet строк таблицы Employees в элемент list1***

***listBox1.Items.Clear();***

***foreach (DataRow row in ds.Tables["Employees"].Rows)***

***{***

***string result = row["id"].ToString() + " " + row["LName"] + " " + row["FName"] + " " + row["MName"];***

***listBox1.Items.Add(result);***

***}***

***}***

***private void btChange\_Click(object sender, EventArgs e)***

***{***

***SqlCommandBuilder builder = new SqlCommandBuilder(adapter);***

***DataTable EmployeeTable = ds.Tables["Employees"];***

***DataRow row = EmployeeTable.NewRow();***

***row["FName"] = "John";***

***row["LName"] = "Smith";***

***EmployeeTable.Rows.Add(row);***

***// builder.GetInsertCommand();***

***adapter.Update(ds, "Employees");***

***}***

В этом примере при нажатии кнопки btChange есь после загрузки данных создается объект SqiCommandBuilder, затем создается новая строка, которая затем добавляется в DataTable. При вызове у адаптера метода Update() происходит анализ изменений, которые произошли. И после этого выполняется соответствующая команда. В данном случае так как идет добавление новой строки, то будет выполняться команда InsertCommand. Однако в данном коде мы нигде явным образом команду не задается, ее автоматически формирует SqlCommandBuilder. Для применения этого класса достаточно вызвать его конструктор, в который передается нужный адаптер:

**Задание на лабораторную работу**

1. Разработать форму (WindowsForms) для отображения одной из таблиц базы данных (из индивидуального задания).

2. Прочитать одну из таблиц базы данных в объект DataSet и отобразить данные таблицы в форме.

3. Создать вручную в DataSet таблицу , внести в нее программно 2-3 записи и отобразить таблицу в форме.

4. Создать в форме элементы для внесения данных в прочитанную из базы данных таблицу, находящуюся в DataSet.

5. Внести в прочитанную из базы данных таблицу, 2- 3 записи и сохранить изменения в базе данных.

6. Заново прочитать таблицу из базы данных и убедиться в том, что внесенные данные сохранились

**Содержание отчета**

1. Тема и цель работы
2. Задание на лабораторную работу
3. Содержимое таблицы базы данных.
4. Код чтение данных из базы данных в DataSet и отображения данных в форме.
5. Внешний вид формы.
6. Код создания и заполнения данными таблицы в DataSet.
7. Внешний вид формы с отображением данных созданной таблицы.
8. Код изменения данных в DataSet и сохранения таблицы в базе данных.
9. Внешний вид формы с отображением данных измененной таблицы.
10. Выводы

**Контрольные вопросы**

1. Для чего предназначен объект DataSet?
2. Какие объекты и методы используются для заполнения объекта DataSet?
3. Как можно создать таблицу в DataSet?
4. Как из объекта DataSet можно отобрать нужные строки из таблицы?
5. Для чего предназначен объект SqlCommandBuilder?
6. Какие объекты и методы используются для передачи данных из DataSet в базу данных?