# Работа с DataAdapter.

**Что такое *DataAdapter***

Объекты *DataAdapter* — являются специфическими объектами провайдеров (то есть для каждого провайдера существуют особые адаптеры данных, например объекты *SqlDataAdapter* и *OleDbDataAdapter*), содержащими такие сведения, как подключение к данным и команды. Эти объекты позволяют вашему приложению соединяться с базой данных и заполнять данными объекты *DataTable* объекта *DataSet,* а также отправлять обновленные данные обратно в базу данных после изменений в данных, сделанных в объектах *DataTable* вашего приложения.

**Создание объектов *DataAdapter***

Вы можете создать объекты *DataAdapter* визуально, используя Data Adapter Configura­tion Wizard, или программно, создав экземпляр желаемого специфического адаптера провайдера и передав конструктору предложение SELECT и допустимый объект *Connection.* Упражнения этого занятия предоставляют детализированные инструкции для создания адаптеров и визуально, и программно.

Следующий код показывает, как создать новый экземпляр *DataAdapter,* передав конструктору исходное предложение SELECT и допустимый объект *Connection:*

// Создание SqlDataAdapter

private SqlDataAdapter SqlDataAdapterl = new SqlDataAdapter ("SELECT \* FROM Shippers", NorthwiridConnection);

**Команды объекта *DataAdapter***

Объекты *DataAdapter* настраиваются обычно указанием одного предложения SE­LECT, использующегося *DataAdapter* для выборки данных и заполнения *DataTable* в *DataSet.* Важно понять, что *DataAdapter* использует это предложение SELECT в качестве основы для автоматического генерирования предложений INSERT, UPDA­TE и DELETE, необходимых для сохранения обратно в базу данных сделанных в *DataSet* изменений, когда вызывается метод *Update* объекта *DataAdapter. DataAdapter* генерирует для вас эти дополнительные команды как часть DataAdapter Configura­tion Wizard (или DataSource Configuration Wizard, в зависимости от вашего метода настройки объектов данных).

При программном создании объектов *DataAdapter* вы можете захотеть создать команды обновления вручную или будете вынуждены это делать при использовании сложных запросов,. Есть два способа настроить команды для *DataAdapter.* создать объекты *Command* и присвоить их соответствующим свойствам *DataAdapter* или же использовать объект *CommandBuilder.*

**Создание объектов *Command* для *DataAdapter***

Создайте команды SELECT, INSERT, UPDATE и DELETE для *DataAdapter*, зако­дировав самостоятельные допустимые команды для вашей конкретной базы данных и назначив их соответствующему свойству команды *DataAdapter.*

Следующий код показывает, как создать самостоятельные команды для *DataAdap­ter* с именем *SqlDataAdapterl.* Этот пример присваивает каждой команде как допус­тимый объект *NorthwindConnection,* так и допустимые предложения SQL (см. пример *CreatingDataAdapterCommands* на диске):

// Создание SqlDataAdapter private SqlDataAdapter SqlDataAdapterl = new SqlDataAdapter("SELECT \* FROM Shippers", NorthwindConnection);

SqlCommand InsertCommand =new SqlCommand("Valid SOL Insert statement", NorthWindConnection);

// настройка требуемых параметров вашей команды...

SqlCommand UpdateCommand =new SqlCommand("Valid SQL Update statement", NorthWindConnection);

// настройка требуемых параметров вашей команды...

SqlCommand DeleteCommand =new SqlCommand("Valid SQL Delete statement", NorthWindConnection);

// настройка требуемых параметров вашей команды...

// Добавление команд к DataAdapter SqlDataAdapterl.InsertCommand = InsertCommand;

Sql1.UpdateCommand = UpdateCommand;

SqlDataAdapterl.DeleteCommand = DeleteCommand;

**Генерирование команд для *DataAdapter* с использованием *CommandBuilder***

Если ваш *DataAdapter* использует предложение SELECT, отбирающее данные из одной таблицы, вы можете применить *CommandBuilder*, чтобы автоматически генерировать для адаптера предложения INSERT, UPDATE и DELETE. Другое требование для *CommandBuilder* — ваше предложение SELECT должно возвратить из таблицы по меньшей мере один столбец, являющийся первичным ключом или уникальным.

Для автоматического генерирования команд с использованием *CommandBuilder* создайте новый экземпляр объекта *CommandBuilder* в коде и укажите в конструкторе *DataAdapter*, для которого вы хотите генерировать команды.

Следующий код показывает, как, используя *CommandBuilder*, генерировать коман­ды для *DataAdapter* с именем *SqlDataAdapterl*:

// Создание экземпляра DataAdapter с допустимым предложением Select SqlDataAdapter SqlDataAdapterl = new SqlDataAdapter ("Valid Single Table Select Statement");

// Создание экземпляра CommandBuilder для SqlDataAdapterl

SqlCommandBuilder commands = new SqlCommandBuilder(SqlDataAdapterl);

## Создание объектов DataAdapter

1. Создайте приложение Windows и назовите его *DataAdapterWizardExample.* В этой версии Visual Studio объекты *DataAdapter* были удалены из Toolbox и заменены

объектами *TableAdapter,* сгенерированными в конструкторе, так что для этого примера сначала добавим *SqlDataAdapter* обратно в Toolbox-

1. Щелкните правой кнопкой мыши раздел Data(Данные) в Toolbox(Панели элементов) и выберите Choose Items(Выберите элементы).
2. В диалоговом окне Choose Toolbox Items(Выбор элементов панели элементов) выберите элемент SqlDataAdapter на вкладке .NET Framework Components и щелкните OK.

Это добавит элемент SqlDataAdapter в Toolbox.

1. Перетащите объект *SqlDataAdapter* в *Form1,* чтобы запустить Data Adapter Con­figuration Wizard.
2. На странице Choose Your Data Connection выберите подключение к базе данных Northwind (или создайте новое подключение в случае необходимости).
3. На странице Choose a Command Туре(Выбор типа команд) оставьте настройку по умолчанию Use SQL Statements(Использовать инструкции SQL) и щелкните Next.
4. На странице Generate the SQL statements(Создание инструкций SQL) введите следующее предложение SQL: SELECT \* FROM SalesLT.Customer
5. Щелкните Finish для завершения мастера и добавления экземпляра настроенного *SqlDataAdapter* к форме.
6. Генерируйте строго типизированный *DataSet,* основанный на настроенном адап­тере, для чего выберите Generate Dataset(Добавить новый источник данных) в меню Project(Проект).

Обратите внимание, что в диалоговом окне выбран *SqlDataAdapter1.* Он содержит сведения, которые будут использоваться для генерирования строго типизирован­ного *DataSet.*

1. Замените в диалоговом окне Generate Dataset имя *DataSet1* на *NorthwindDataSet.* Щелкните OK для создания нового *DataSet* и добавления его к проекту.
2. Перетащите *DataGridView* в *Forml.*
3. Дважды щелкните пустую область формы для создания обработчика события *Form1Load.*
4. Добавьте к обработчику события *Forml Load* следующий код:

// Настройка сетки для отображения таблицы клиентов

dataGridView1.DataSource = northwindDataSet1.Customer;

// Вызов метода Fill объекта DataAdapter для загрузки таблицы

// клиентов данными

sqlDataAdapter1.Fill(northwindDataSet1.Customer);

Выполните приложение и убедитесь, что таблица *Customer* появляется в сетке.

1. Добавьте к форме кнопку и установите следующие свойства:

* *Name* - *UpdateButton*
* *Text* - *Save Changes*

1. Дважды щелкните кнопку Save Changes для создания обработчика события щелчка кнопки.
2. Добавьте к обработчику события *UpdateButtonjClick* следующий код:

// Вызов метода адаптера данных Update для сохранения изменений

// в базе данных

SqlDataAdapterl.Update(northwindDataSet1);

Объекты *DataAdapter* вызывают события, которые могут использоваться при вы­полнении операций по отношению к источнику данных, например события *Row- Updating, RowUpdated* и *FillError.* Последующие шаги добавляют к примеру код и логику, которые демонстрируют работу с этими событиями.

1. В списке Class Name (левый раскрывающийся список вверху редактора кода) вы­берите *SqlDataAdapterl* (для создания события в проекте C# выберите *SqlData­Adapterl* в области компонентов и используйте кнопку Events в окне Properties).
2. Создайте обработчик для события *RowUpdating,* выбрав *RowUpdating* из раскры­вающегося списка Method Name.
3. Добавьте к обработчику события *RowUpdating* следующий код:

// Создание CustomerRow и присвоение ему изменяемой строки NorthwindDataSet.CustomersRow CustRow =

(NorthwindDataSet.CustomersRow)e.Row;

// Отображение диалогового окна для подтверждения обновления

DialogResult response = MessageBox.Show("Continue updating " +

CustRow.CustomerlD.ToString() + "?", "Continue Update?", MessageBoxButtons.YesNo);

// Отмена обновления, если пользователь выбрал Нет

if (response == DialogResult.No)

{

e.Status = UpdateStatus.SkipCurrentRow;

}

1. Создайте обработчик для события *RowUpdated,* выбрав *RowUpdated* из раскры­вающегося списка Method Name (для создания события в проекте C# выберите *SqlDataAdapterl* в области компонентов и используйте кнопку Events в окне Pro­perties).
2. Добавьте к обработчику события *RowUpdated* следующий код:

// Создание CustomerRow и присвоение ему изменяемой строки NorthwindDataSet.CustomersRow CustRow =

(NorthwindDataSet.CustomersRow)e.Row; MessageBox.Show(CustRow.CustomerID.ToString() + ” has been updated");

// Сброс таблицы после обновления строки для отражения изменений northwindDataSetl.Customers.Clear();

SqlDataAdapterl.Fill(northwindDataSet1.Customers);

1. Создайте обработчик для события *FillError,* выбрав *FillError* из раскрывающегося списка Method Name (для создания события в проекте C# выберите *SqlDataAdap­terl* в области компонентов и используйте кнопку Events в окне Properties).
2. Добавьте к обработчику события *FillError* следующий код:

// Отображение диалогового окна для реагирования на ошибку

DialogResult response = MessageBox.Show

("The following error occurred while Filling the DataSet: " + e.Errors.Message.ToStringO + " Continue attempting to fill?",

“FillError Encountered“, MessageBoxButtons.YesNo);

// Попытка продолжения, если пользователь выбрал Да

if (response == DialogResult.Yes)

{

e.Continue = true;

}

else

{

e.Continue = false;

}

1. Выполните приложение и измените несколько значений по меньшей мере в двух различных записях.
2. Щелкните кнопку Save Changes.

В этот момент события адаптера данных позволяют вам реагировать в диалоговом окне на запрос о необходимости продолжать обновление. Обратите внимание, что вы можете индивидуально обработать каждую конкретную строку, которую адаптер пытается обновить.

## Создание объектов DataAdapter в коде

1. Создайте приложение Windows и назовите его *ProgrammingDataAdapters.*
2. Добавьте к форме *DataGridView.*
3. Добавьте к форме пространство имен *System.Data.SqlClient.*
4. Добавьте объект *Connection* для базы данных AdventureWorksLT.
5. В этот момент код вашей формы должен быть похож на следующий:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms; using System.Data SqlClient;

namespace Prog rammingDataAdaptersCS

{

public partial class Forml : Form

{

public Forml()

{

InitializeComponent();

}

private SqlConnection NorthwindConnection = new SqlConnection("Data Source=" +

".\\sqlexpress;Initial Catalog=Northwind;Integrated Security=True");

}

}

1. Добавьте ниже *NorthwindConnection* код для создания *DataAdapter.*

private SqlDataAdapter SqlDataAdapterl;

1. Поскольку объекты *DataAdapter* используются для заполнения объектов ***DataSet,*** ниже предыдущих строк добавьте код для создания нового *NorthwindDataSet* с таб­лицей Customer:

private DataSet NorthwindDataSet = new DataSet("Northwind"); private DataTable CustomersTable = new DataTable("Customer");

1. Создайте обработчик события *Forml Load* и добавьте к нему следующий код-

SqlDataAdapterl = new SqlDataAdapter("SELECT \* FROM Customer", NorthwindConnection):

NorthwindDataSet.Tables.Add(CustomersTable);

SqlDataAdapterl.Fill(NorthwindDataSet.Tables["Customer"]); dataGridViewl.DataSou rce = NorthwindDataSet.Tables["Customer"];

В этот момент *DataAdapter* содержит только команду SELECT. Создадим *Сот mandBuilder,* который предоставит дополнительные команды INSERT, UPDATE и DELETE, требующиеся для обновления базы данных при вызове метода *Data­Adapter. Update.*

1. Добавьте в обработчик события *Form1\_Load* ниже предварительно добавленных строк следующий код:

SqlCommandBuilder commands = new SqlCommandBuilder(SqlDataAdapterl);

1. Добавьте к форме кнопку и установите следующие свойства:

* *Name* - *UpdateButton*
* *Text* = *Save Changes*

1. Дважды щелкните кнопку Save Changes и добавьте к обработчику события *Upda­teButtonjClick* следующий код:

No rthwindDataSet.Endlnit();

SqlDataAdapterl.Update(NorthwindDataSet.Tables["Customer"]);

1. Выполните приложение и измените в сетке значение в одной из записей
2. Щелкните кнопку Save Changes для отправки нового значения в базу данных.
3. Убедитесь, что изменения сохранены в базе данных.

**Контрольные вопросы**

* + - 1. Какие основные объекты составляют *DataAdapter?*
  1. Объекты *Connection* и *DataTable.* Б. Объекты *Connection* и *Command.*
  2. Объекты *DataTable* и *DataSet* Г. Объекты *Command* и *DataSet*
     + 1. *2.* Как настроить *DataAdapter* для сохранения изменений обратно в базу данных (выберите все, что применяется)?

1. Создать экземпляр объекта *CommandBuilder* и передать его *DataAdapter* с на­строенной командой SELECT.
2. Вызвать метод DataAdapter.Update.
3. Присвоить допустимый объект Command свойствам InsertCommand, UpdateCom-mand и DeleteCommand объекта DataAdapter.
4. Вызвать метод DataAdapter.Fill.