Крос-платформне програмування

Лекція 9

Формальні та візуальні методи конструювання компонентів. Технологія ADO.NET

16 квітня, 2014

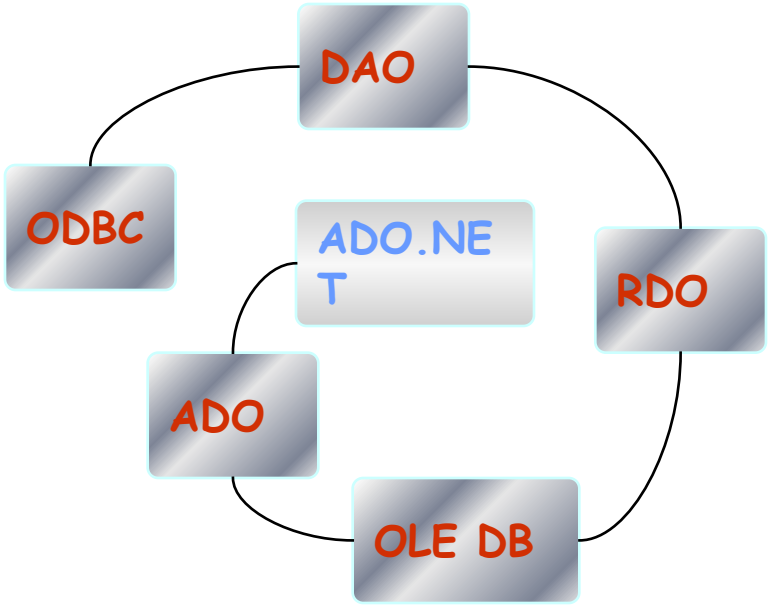
Примітка: слайди лекції підготовлені за матеріалами http://jskreator.narod.ru/proaspnet20cs/glance.htm

План наступих лекцій

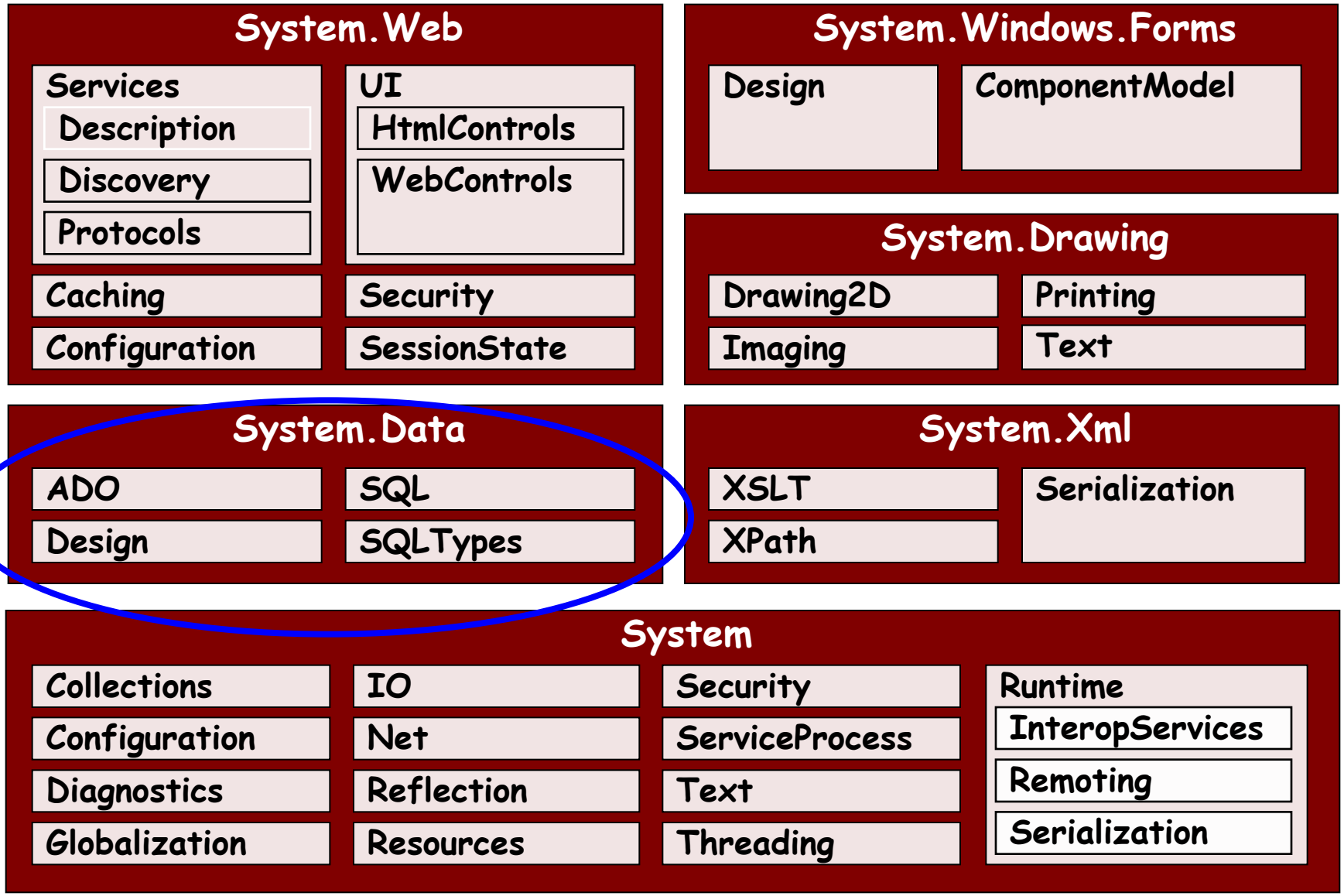
- Основи технології доступу до даних ADO.NET. Архітектура ADO.NET: набір даних та провайдер даних
- Об'єктна модель ADO.NET. Робота з БД у приєднаному режимі. Класи Connection, Command, DataReader, Transaction
- Робота з БД у від'єднаному режимі. Класи DataSet, DataReader, Tables та Relationships
- Основи прив'язки даних
- Розширені елементи керування даними GridView, DetailsView, FormView. Елементи керування джерелами даний

Технології Microsoft для роботи з БД

- ODBC API (Open Database Connectivity)
- RDO (Remote Data Objects)
- DAO (Data Access Objects)
- OLE DB (Object Linking and Embedding, Database)
- ADO (ActiveX Data Objects)
- ADO.NET



Простори імен FCL



Провайдер даних

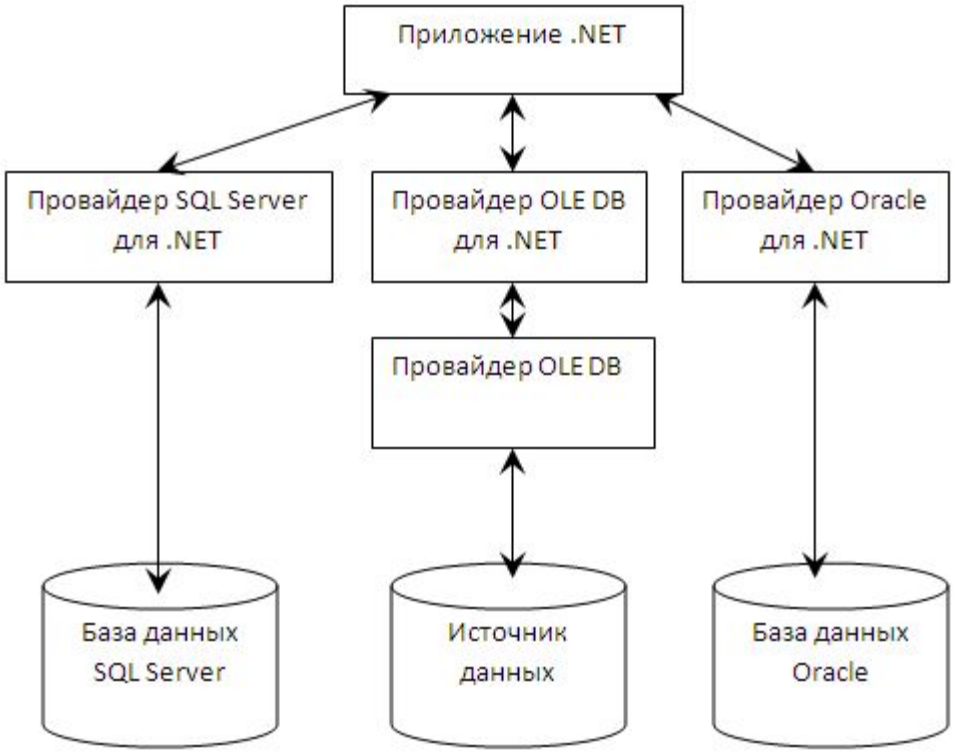
- **Провайдер даних** *(data provider)* – набір класів ADO.NET, що дозволяють отримувати доступ до БД, виконувати команди SQL та отримувати дані
- Складається з набору класів
  - Connection – забезпечує підключення до БД
  - Command – виконує команди SQL або збережені процедури
  - DataReader – надає доступний лише для однонаправленого читання набір записів, подключений до БД
  - DataAdapter – заповнює від'єднаний об'єкт DataSet або DataTable та оновлює його вміст
- Назви класів провайдера містять префікс перед назвою типу класу
  - OleDb<ім'яКласу> - для провайдера OleDb
  - Sql<ім'яКласу> - для провайдера SqlClient

Провайдери даних Microsoft ADO.NET

Провайдери даних	Простір імен	Опис
ODBC	System.Data.Odbc	Підключення до більшості драйверів ODBC, зокрема **OdbcCommand**, **OdbcConnection** та **OdbcDataAdapter**
OLE DB	System.Data.OleDb	Підключення до провайдера OLE DB, зокрема **OleDbCommand**, **OleDbConnection** та **OleDbDataAdapter**
SQL Server	System.Data.SqlClient	Підключення до БД Microsoft SQL Server, у тому числі **SqlDbCommand**, **SqlDbConnection** та **SqlDbDataAdapter**
Oracle	System.Data.OracleClient	Підключення до БД Oracle, зокрема **OracleCommand**, **OracleConnection** та **OracleDataAdapter**

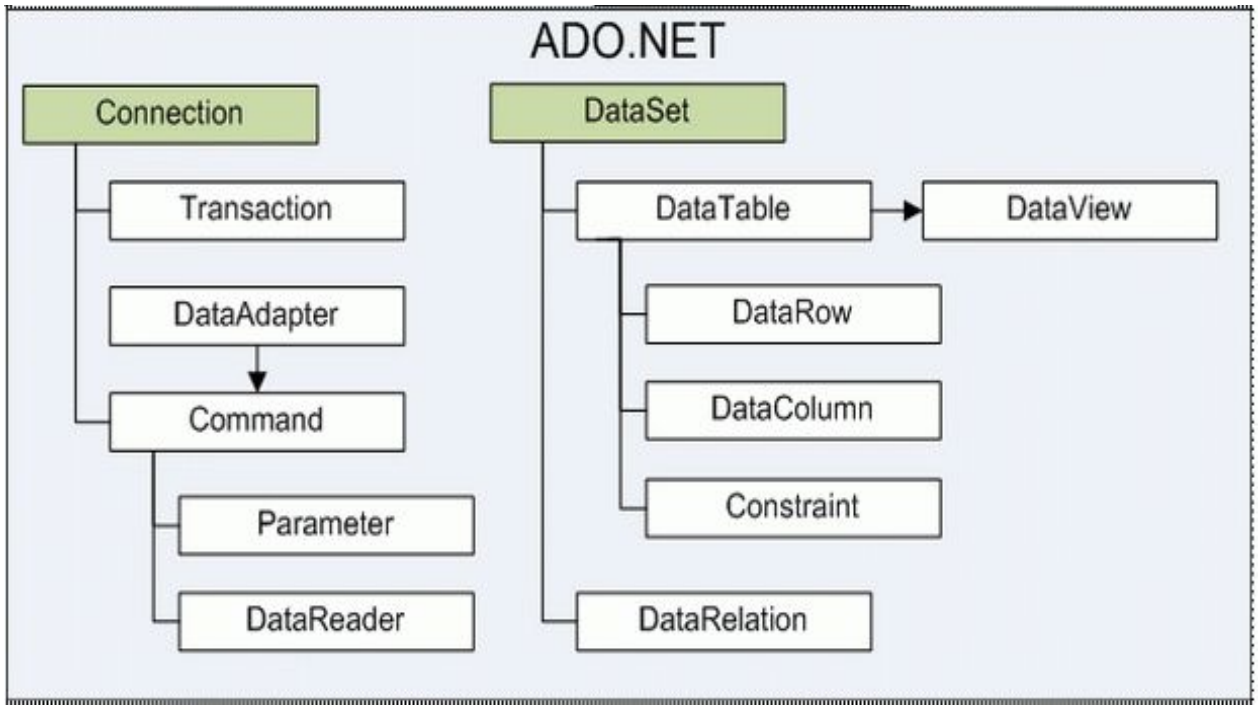
Архітектура ADO.NET

- **Переваги моделі провайдерів**
  - **Розширюваність моделі – можна створювати власні провайдери**
  - **Кожен провайдер може використовувати відповідну оптимізацію та додавати нестандартні засоби**



Типи об'єктів ADO.NET

- **Об'єкти, засновані на з'єднанні** – об'єкти провайдера даних, що виконують SQL-оператори, підключаються до БД та заповнюють DataSet
  - Connection, Command та DataReader
- **Об'єкти, засновані на вмісті** – об'єкти, що у дійсності лише "упаковують" дані
  - DataSet, DataColumn, DataRow, DataRelation тощо



Способи роботи з БД

- **Приєднаний режим** або з підтримкою з'єднання *(Connected,* Forward-only, read-only)
  - Програма встановлює з'єднання з БД, виконує запит, читає і обробляє результати, закриває з'єднання з БД
  - Використовується об'єкт DataReader
- **Від'єднаний** або з розривом з'єднання *(Disconnected)*
  - Програма встановлює з'єднання з БД, виконує запит, читає і зберігає результати у об'єкті DataSet для подальшої обробки, від'єднується від БД
  - Виконується робота з даними (додавання, зміна, видалення)
  - Мінімізується час з'єднання з базою даних
  - Використовується об'єкт DataAdapter

Шаблон роботи з БД у приєднаному режимі

```
1.) Оголосити з'єднання
try {
    1.) Відкрити з БД
```

```
    2.) Створити та виконати команди
```

```
    3.) Обробити результати
```

```
    4.) Звільнити ресурси
} catch ( Exception ) {
    Handle exception
} finally {
    try {
        4.) Закрити з'єднання
    } catch (Exception)
        { Handle exception }
}
```

Клас Connection

- виконує реальний обмін даними між БД та застосуванням
- є частиною Data Provider
- властивості
  - ConnectionString – рядок з'єднання
  - ConnectionTimeout - час очікування при спробі установки підключення
  - DataBase – ім'я поточної БД або БД, яка використовуватиметься після відкриття з'єднання
- методи
  - Open() – відкрити з'єднання
  - Close() – закрити з'єднання

Рядок з'єднання

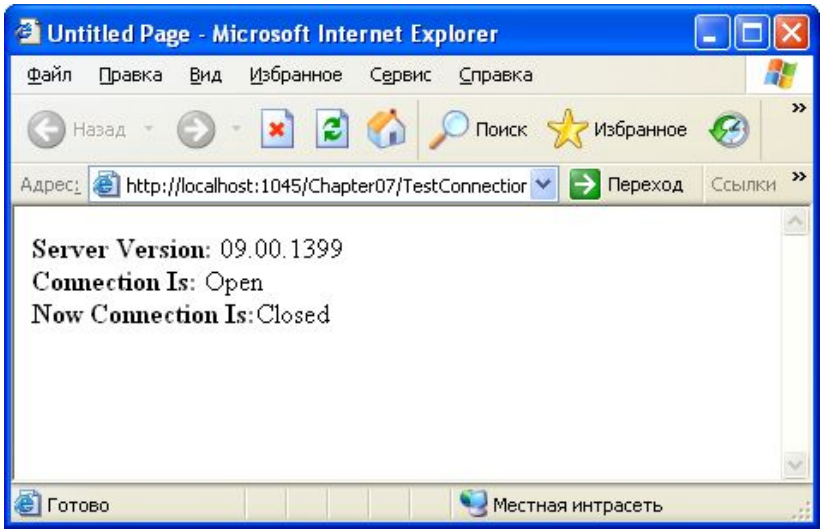
- **Рядок з'єднання** – серія розділених крапкою з комою пар "ім'я-значення", що специфікують базову інформацію, необхідну для встановлення з'єднання
- Основні параметри рядка з'єднання
  - `Data Source=(local)\SQLEXPRESS;` - сервер, на якому знаходиться БД
    - » (local), localhost, . (просто точка)
  - `Initial Catalog=<ім'я БД>;` - БД, яку слід використовувати
  - `IntegratedSecurity=True;` - інтегрована безпека
    - » True, SSPI, yes
  - `user id=<ідентифікатор>;` - ім'я користувача
  - `pwd=<пароль>;` - пароль користувача
  - `Provider=<провайдер>` - налаштування, що ідентифікують драйвер (для ODBC та OLEDB)
  - тощо

Приклади рядків з'єднання

```
string connectionString =
"Data Source=localhost;Initial Catalog=Northwind;" +
"Integrated Security=SSPI";

string connectionString =
"Data Source=localhost;Initial Catalog=Northwind;" +
"user id=sa; password=opensesame";

string connectionString = "Data Source=localhost;Initial Catalog=Sales;" +
"user id=sa;password=;Provider=MSDAORA";

string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" +
@"Data Source=C:\DataSources\Northwind.mdb";
```

Приклад використання об'єкта Connection

```
// Створити об'єкт Connection
string connectionString =
   "Data Source=localhost;Initial Catalog=Northwind;" +
   "Integrated Security=SSPI";
SqlConnection con = new SqlConnection(connectionString);
try
{
   // Відкрити з'єднання
   con.Open();
   lblInfo.Text = "<b>Server Version:</b> " + con.ServerVersion;
   lblInfo.Text += "<br /><b>Connection Is:</b> " + con.State.ToString();
}
catch (Exception err)
{
   // Обробка помилки з відображенням інформації
   lblInfo.Text = "Error reading the database. ";
   lblInfo.Text += err.Message;
}
finally
{
   con.Close();
   lblInfo.Text +=
       "<br /><b>Now Connection Is:</b>";
   lblInfo.Text += con.State.ToString();
}
```



Клас Command

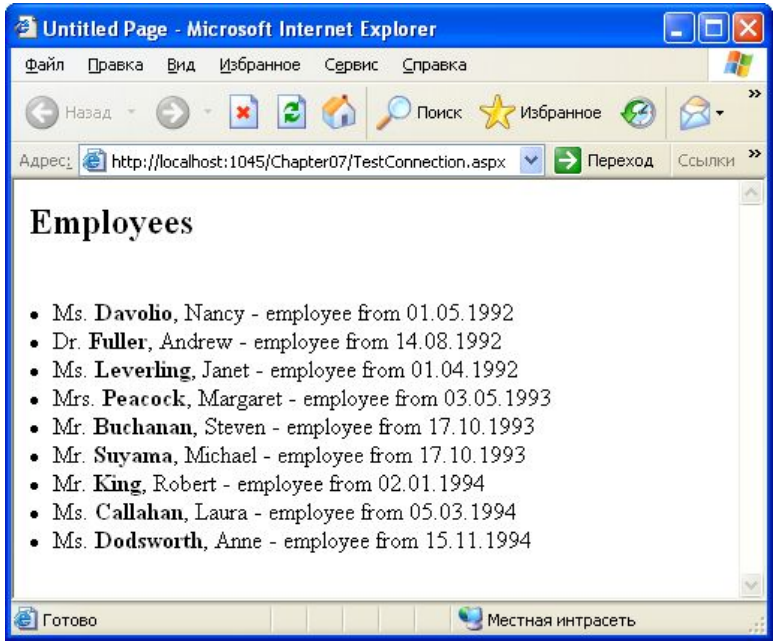
- Клас **Command** дозволяє виконувати дії з БД (вибірку, оновлення, доповнення, видалення тощо)
- Властивості
  - **CommandType** – встановлює один з трьох типів команд
    - » `CommandType.Text` - оператори SQL
    - » `CommandType.TableDirect` – робота з конкретною таблицею
    - » `CommandType.StoredProcedure` – виклик збереженої у БД процедури
  - **CommandText** містить:
    - » текст оператора SQL (для типу `CommandType.Text`)
    - » ім'я таблиці (для типу `CommandType.TableDirect`)
    - » ім'я збереженої процедури з параметрами (для типу `CommandType.StoredProcedure`)
  - **Connection** – посилання на відкрите з'єднання (об'єкт `Connection`)
  - **Parameters** – колекція параметрів запиту

Основні методи класу Command

- **ExecuteReader()** – виконує оператор SELECT, створює та повертає посилання на об'єкт DataReader, який містить результат виконання запиту
- **ExecuteNonQuery()** – виконує оператори INSERT, DELETE, UPDATE на мові SQL (повертає кількість оброблених записів)
- **ExecuteScalar()** – повертає перший рядок першого стовбця у результуючому наборі (використовуючи функції COUNT, AVG, MIN, MAX, SUM тощо)

Приклад виклику методу ExecuteReader()

```
string connectionString =
WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
string sql = "SELECT * FROM Employees";
SqlCommand cmd = new SqlCommand(sql, con);
con.Open();

SqlDataReader reader = cmd.ExecuteReader();
StringBuilder htmlStr = new StringBuilder("");
while (reader.Read())
{
   htmlStr.Append("<li>");
   htmlStr.Append(reader["TitleOfCourtesy"]);
   htmlStr.Append(" <b>");
   htmlStr.Append(reader.GetString(1));
   htmlStr.Append("</b>, ");
   htmlStr.Append(reader.GetString(2));
   htmlStr.Append(" - employee from ");
   htmlStr.Append(reader.GetDateTime(6).ToString("d"));
   htmlStr.Append("</li>");
}
reader.Close();
con.Close();
HtmlContent.Text = htmlStr.ToString();
```

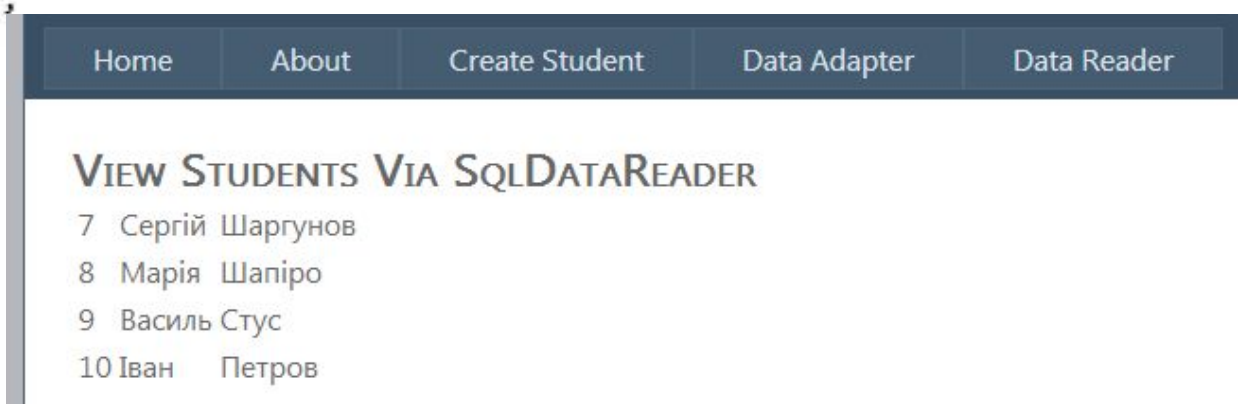


Клас DataReader

- Дозволяє послідовно **читати** записи, отримані з БД за допомогою об'єкта Command, **тільки в одному напрямку** (від початку до кінця)
- Одночасно надає доступ тільки до одного запису вибірки
- Методи
  - **Read()** - читає поточний запис та переміщує курсор на наступний запис (повертає true, якщо наступний запис прочитано, інакше повертає false)
  - **Close()** - закінчення роботи з даними в DataReader
  - **GetValue()** – повертає значення поля за вказаним ім'ям або індексом
- Значення поля у записі можна визначити, використовуючи индексатор
  - `dr[n]` або `dr["ім'я_поля"]`

Приклад: застосування SqlDataReader

```
SqlCommand cmd = con.CreateCommand();
cmd.CommandText = @"SELECT Id, FirstName, LastName FROM Student";
using (SqlDataReader r = cmd.ExecuteReader())
{
    while (r.Read())
    {
        TableRow tr = new TableRow();
        TableCell cell;

        cell = new TableCell();
        cell.Text = r[0].ToString();
        tr.Cells.Add(cell);

        cell = new TableCell();
        cell.Text = r[1].ToString();
        tr.Cells.Add(cell);

        cell = new TableCell();
        cell.Text = r[2].ToString();
        tr.Cells.Add(cell);

        tableStudents.Rows.Add(tr);
    }
}
```



Метод ExecuteNonQuery()

- **Дозволяє виконувати**
  - команди корегування (повертає кількість змінених записів) - INSERT, UPDATE, DELETE

```
INSERT INTO tbl (f1, f2, f3) VALUES ('xxx', 1986, 'yyy')
UPDATE child SET id = 27 WHERE year = 1997
DELETE FROM child WHERE ID = 5
```

  - інші команди, які не повертають значень (результат -1) - CREATE DATABASE, CREATE TABLE

- **Приклад**

```
SqlCommand cmd = con.CreateCommand();
cmd.CommandText = "INSERT INTO Students(FirstName, LastName)" +
"VALUES('Іван', 'Петров')";
cmd.ExecuteNonQuery();
```

FirstName	LastName
Сергій	Шаргунов
Марія	Шапіро
Василь	Стус
Іван	Петров

Приклад: виклик методу SqlCommand.ExecuteScalar()

```
cmd.CommandText = "SELECT COUNT(Id) FROM Students";
int count = (int)cmd.ExecuteScalar();
Console.WriteLine("Count is: " + count);
```

Атаки впровадженням SQL

- **Впровадження SQL** (SQL injection) – процес передачі SQL-коду застосуванню у такий спосіб, який не передбачений його розробником
  - можливе через поганий дизайн програми, напр., при побудові SQL-рядків для створення команд з уведеними користувачем значеннями

```
string connectionString =
  WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
string sql =
  "SELECT Orders.CustomerID, Orders.OrderID, COUNT(UnitPrice) AS Items, " +
  "SUM(UnitPrice * Quantity) AS Total FROM Orders " +
  "INNER JOIN [Order Details] " +
  "ON Orders.OrderID = [Order Details].OrderID " +
  "WHERE Orders.CustomerID = ' " + txtID.Text + "' " +
  "GROUP BY Orders.OrderID, Orders.CustomerID";
SqlCommand cmd = new SqlCommand(sql, con);

con.Open();
SqlDataReader reader = cmd.ExecuteReader();
...
```

Enter Customer ID:

ALFKI | Get Records

CustomerID	OrderID	Items	Total
ALFKI	10643	3	1086,0000
ALFKI	10692	1	878,0000
ALFKI	10702	2	330,0000
ALFKI	10835	2	851,0000
ALFKI	10952	2	491,2000
ALFKI	11011	2	960,0000

Атаки впровадженням SQL (2)

- Припустимо у текстове поле уведений текст ALFKI' OR '1' = '1
- Буде отримано запит

```
SELECT Orders.CustomerID, Orders.OrderID,
   COUNT(UnitPrice) AS Items,
   SUM(UnitPrice * Quantity) AS Total
FROM Orders INNER JOIN [Order Details] ON Orders.OrderID -
   [Order Details].OrderID
WHERE Orders.CustomerID = 'ALFKI' OR '1' = '1'
   GROUP BY Orders.OrderID, Orders.CustomerID
```

- Умова 1=1 істинна для усіх рядків, тому замість інформації про поточного замовника, буде виведена інформація про усі замовлення

CustomerID	OrderID	Items	Total
VINET	10248	3	440,0000
TOMSP	10249	2	1863,4000
HANAR	10250	3	1813,0000
VICTE	10251	3	670,8000
SUPRD	10252	3	3730,0000
HANAR	10253	3	1444,8000
CHOPS	10254	3	625,2000
RICSU	10255	4	2490,5000
WELLI	10256	2	517,8000
HILAA	10257	3	1119,9000
ERNSH	10258	3	2018,6000

Параметрезовані команди

- **Параметрезована команда** - команда, що використовує символи-заповнювачі (placeholder parameter) у тексті SQL
  - Дозволяють міняти SQL запит без переписування його тексту
  - Використовуються при виклику збереженої процедури для передачі вхідних даних та отримання результатів
- Заповнювач вказує місце для динамічно застосовуваних значень
  - Для Odbc поля параметра задаються символами "?"

```
select EmpId, Title, FirstName, LastName from Employees
where (FirstName = ?, LastName = ? )
```

  - Для OleDbCommand та SqlCommand використовуються іменовані поля параметрів - @Xxxxx

```
select EmpId, Title, FirstName, LastName from Employees
where (FirstName = @First, LastName = @Last )
```

Додавання параметрів

- Властивості класу xxxParameter для опису параметрів запиту
  - ParameterName - ім'я xxxParameter
  - xxxType (напр., SqlDbType) - тип даних параметра, що представляється у вигляді CLR-типу
  - Direction - вид параметра (тільки для введення, тільки для виведення, для введення і для виведення або параметр для повернення значення)
  - Value - значення параметра
- В об'єкті Command є колекція параметрів (об'єктів Parameter) Parameters
- Для використання параметра необхідно створити об'єкт Parameter та зберегти його у колекції Parameters
- Методи додавання
  - Add(parameter), AddWithValue(name, value)

Приклад: опис параметру

- Варіант попереднього запиту (за ідентифікатором замовника відобразити перелік його замовлень), що виключає можливість атаки впровадженням

```
string connectionString =
    WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
string sql =
    "SELECT Orders.CustomerID, Orders.OrderID, COUNT(UnitPrice) AS Items,"+
    "SUM(UnitPrice * Quantity) AS Total FROM Orders " +
    "INNER JOIN [Order Details] " +
    "ON Orders.OrderID = [Order Details].OrderID " +
    "WHERE Orders.CustomerID = @CustID " +
    "GROUP BY Orders.OrderID, Orders .CustomerID";
SqlCommand cmd = new SqlCommand(sql, con);
cmd.Parameters.Add("@CustID", txtID.Text);
con.Open();
SqlDataReader reader = cmd.ExecuteReader();
...
```

CustomerID	OrderID	Items	Total
ALFKI	10643	3	1086,0000
ALFKI	10692	1	878,0000
ALFKI	10702	2	330,0000
ALFKI	10835	2	851,0000
ALFKI	10952	2	491,2000
ALFKI	11011	2	960,0000

Крос-платформне програмування

Лекція 10

Технологія ADO.NET. Автономний режим доступу до даних

23 квітня, 2014

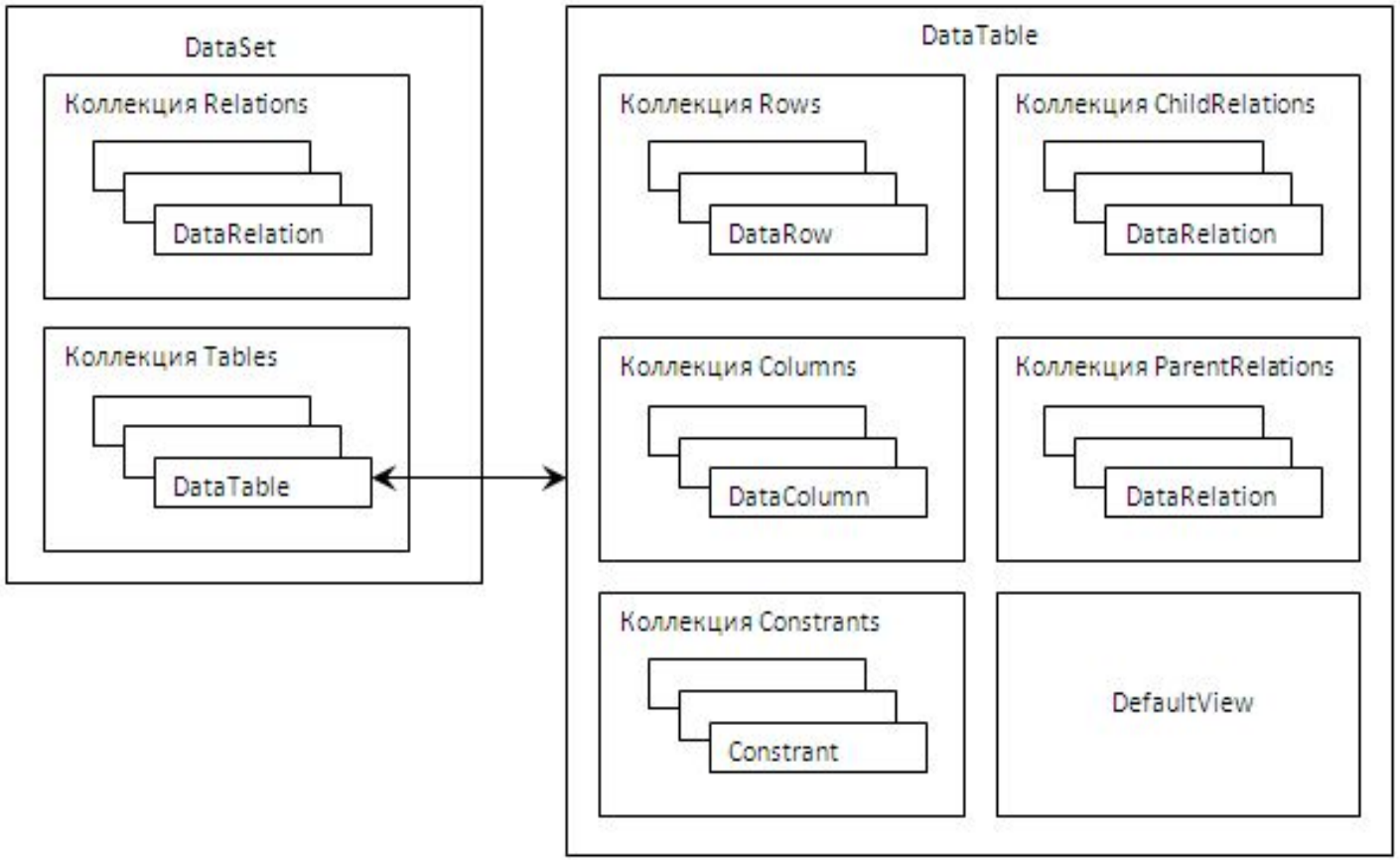
Примітка: слайди лекції підготовлені за матеріалами http://jskreator.narod.ru/proaspnet20cs/glance.htm

Коли DataSet краще, ніж DataReader

- потрібен зручний пакет для відправки даних іншому компоненту
- потрібен зручний формат файлу для серіалізації даних на диск
- потрібно організувати навігацію у двох напрямках по великому об'єму даних
- потрібно виконувати навігацію по декількох різних таблицях
- потрібно використовувати прив'язку даних до елементів керування користувацького інтерфейсу
- необхідно маніпулювати такими даними, як XML
- необхідно виконувати пакетні оновлення через веб-службу

Клас DataSet

- **Властивості**
  - **`Tables` - колекція з нуля або більше таблиць**
  - **`Relationships` - колекція з нуля або більше відношень, які можна застосовувати для зв'язування таблиць між собою**

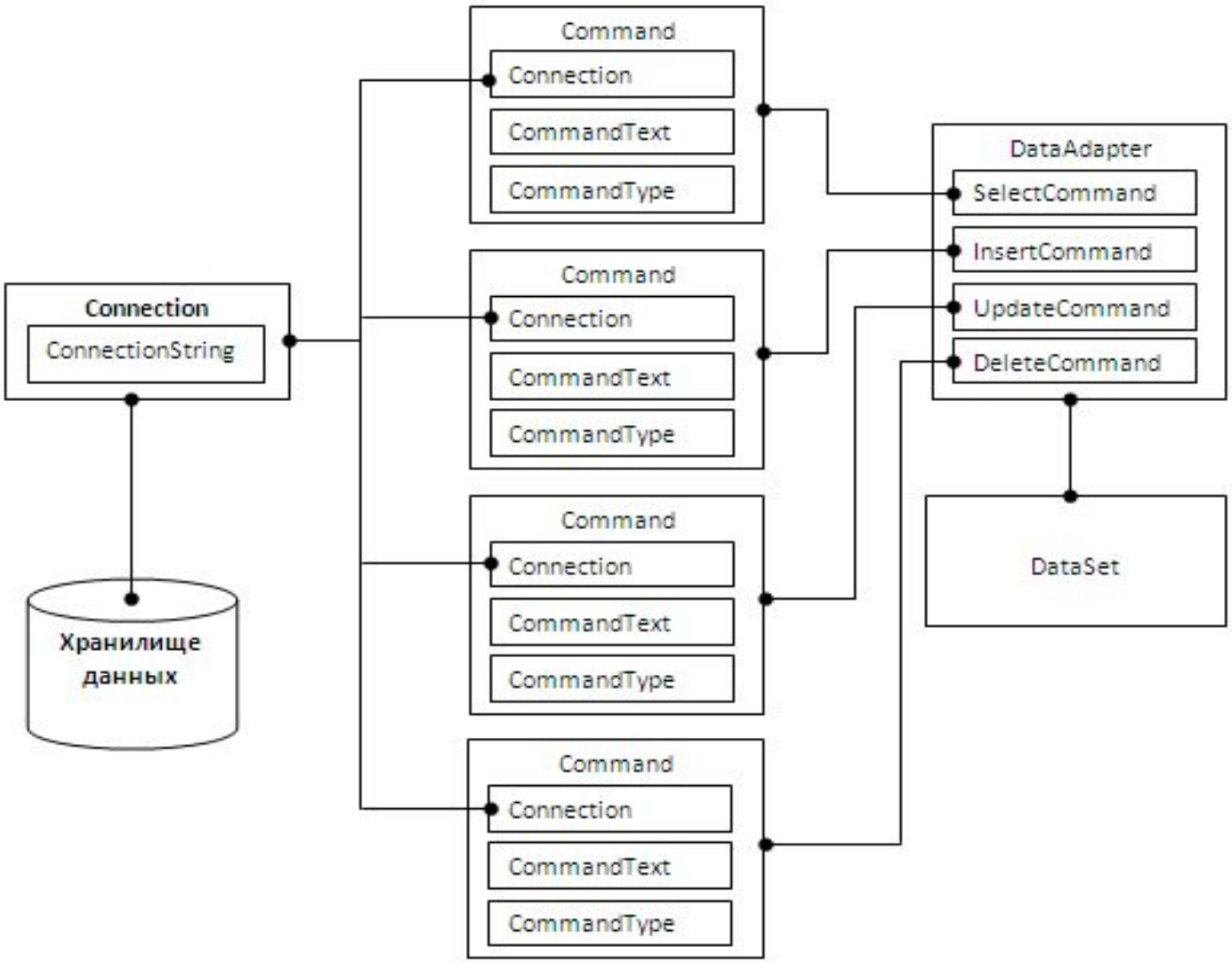


Клас DataAdapter

- **DataAdapter – забезпечує доступ до від'єднаних даних**
  - **Посередник між БД та об'єктом DataSet**
  - **Включає усі доступні команди для виконання запитів та оновлення джерела даних**
- **Властивості**
  - **SelectCommand, InsertCommand, DeleteCommand, UpdateCommand**
- **Ключові методи**

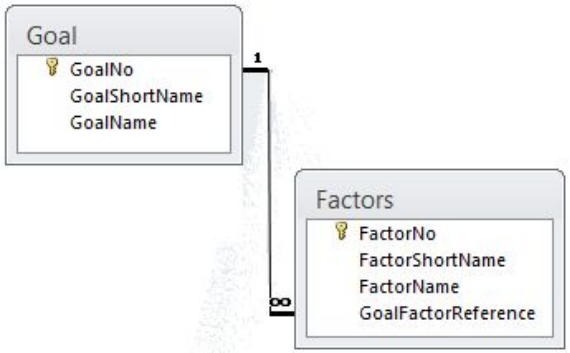
Метод	Опис
Fill()	Додає **DataTable** до **DataSet** за рахунок виконання запиту в **SelectCommand**
Update()	Перевіряє всі зміни в окремій **DataTable** і застосовує пакет цих змін до джерела даних за допомогою виконання відповідних операцій **InsertCommand**, **UpdateCommand** та **DeleteCommand**

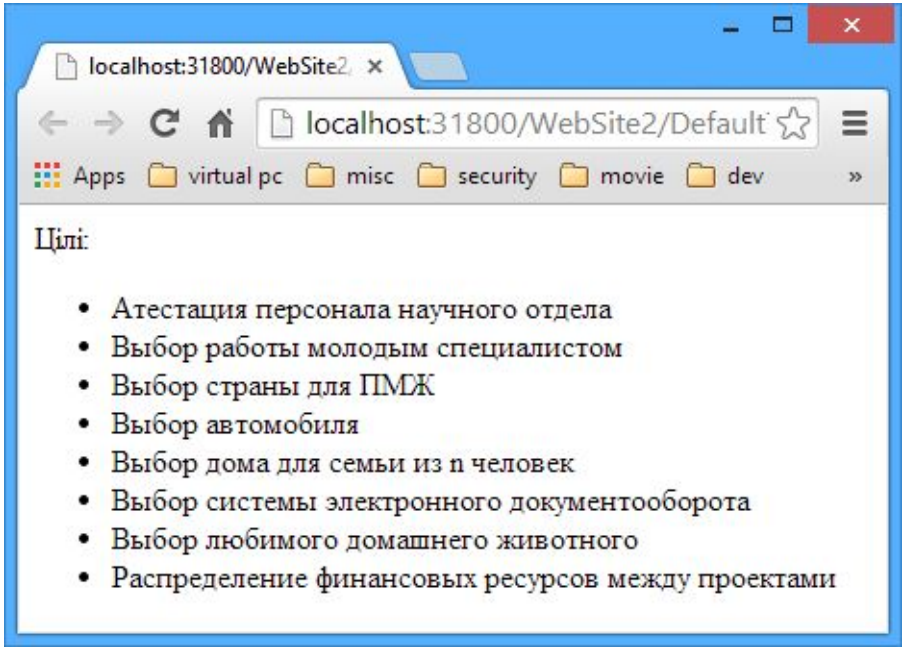
Взаємодія DataAdapter з джерелом даних



Приклад: наповнення DataSet

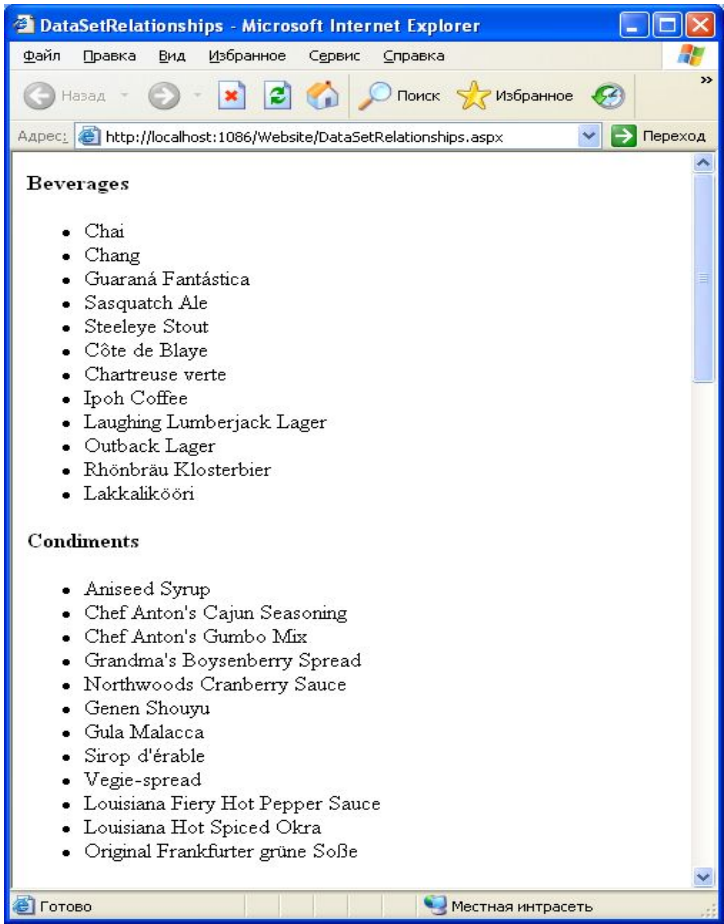
```
// Створення з'єднання
string connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" + @"Data Source=g:\nnn\db1.mdb";
OleDbConnection con = new OleDbConnection(connectionString);
try
{   // Відкрити з'єднання.
    con.Open();
    // Визначення тексту запиту
    string sql = "SELECT * FROM Goal";
    // Створення екземпляру класу DataAdapter
    OleDbDataAdapter da = new OleDbDataAdapter(sql, con);

    // Створення пустого DataSet
    DataSet ds = new DataSet();
    // Наповнити DataSet даними з таблиці Factors
    da.Fill(ds, "Goal");

    // Пройти по усіх цілях та згенерувати перелік
    StringBuilder htmlStr = new StringBuilder("");
    htmlStr.Append("<ul>");
    foreach (DataRow row in ds.Tables["Goal"].Rows)
    {
        htmlStr.Append("<li>");
        htmlStr.Append(row["GoalName"].ToString());
        htmlStr.Append("</li>");
    }
    htmlStr.Append("</ul>");
    HtmlContent.Text = htmlStr.ToString();
}
```





Робота з множинними таблицями та відношеннями

```
string connectionString =
    WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
string sqlCat = "SELECT CategoryID, CategoryName FROM Categories";
string sqlProd = "SELECT ProductName, CategoryID FROM Products";
SqlDataAdapter da = new SqlDataAdapter(sqlCat, con);
DataSet ds = new DataSet();
try
{
   con.Open();
   // Наповнити DataSet даними з таблиці Categories
   da.Fill(ds, "Categories");
   // Змінити текст команди та отримати дані
   // таблиці Products
   da.SelectCommand.CommandText = sqlProd;
   da.Fill(ds, "Products");
}
finally
{  con.Close();
}
// Визначити відношення між Categories та Products
DataRelation relat = new DataRelation("CatProds",
   ds.Tables["Categories"].Columns["CategoryID"],
   ds.Tables["Products"].Columns["CategoryID"]);
// Додати відношення до DataSet.
ds.Relations.Add(relat);
// Продовження див. на наступному слайді
```



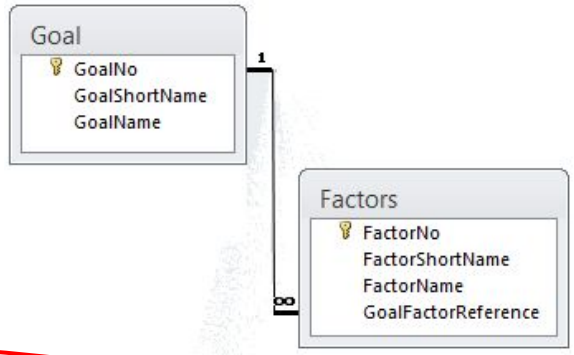
Робота з множинними таблицями та відношеннями (2)

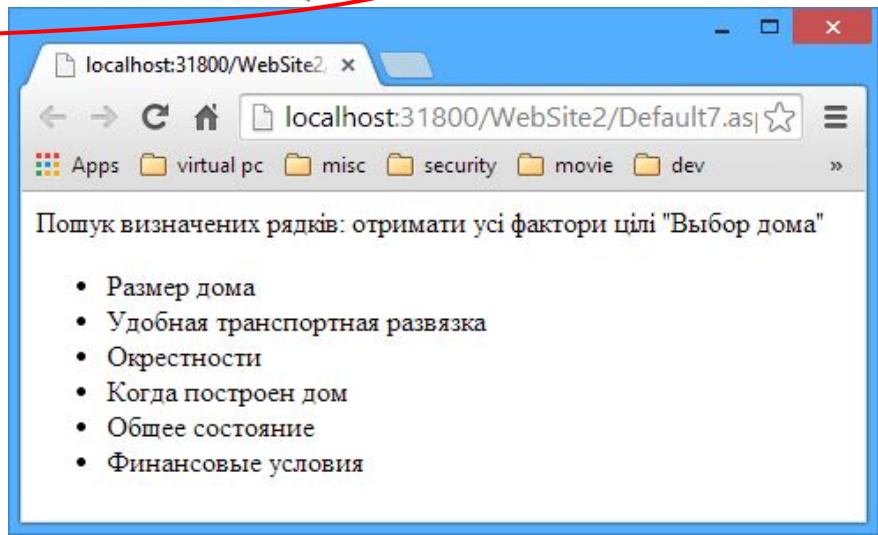
```
// Відобразити дані у формі
StringBuilder htmlStr = new StringBuilder("");
// Пройти у циклі по всіх записах категоріях та побудувати строку HTML.
foreach (DataRow row in ds.Tables["Categories"].Rows)
{    htmlStr.Append("<b>");
     htmlStr.Append(row["CategoryName"].ToString());
     htmlStr.Append("</b><ul>");
     // Получить дочерние (products) записи для родителя (category).
     DataRow[] childRows = row.GetChildRows(relat);
     // Пройти по всем продуктам данной категории.
     foreach (DataRow childRow in childRows)
     {
     htmlStr.Append("<li>");
     htmlStr.Append(childRow["ProductName"].ToString());
     htmlStr.Append("</1i>");
     }
     htmlStr.Append("</ul>");
}
HtmlContent.Text = htmlStr.ToString();
```

Пошук визначених рядків

- **Метод Select() класу DataTable дозволяє отримувати масив об'єктів DataRow на основі SQL-виразу**
  - **Приклад: отримати усі фактори цілі "Выбор дома"**

```
// Визначення тексту запиту
string sql = "SELECT * FROM Factors";
// Створення екземпляру класу DataAdapter
OleDbDataAdapter da = new OleDbDataAdapter(sql, con);

// Створення пустого DataSet
DataSet ds = new DataSet();
// Наповнити DataSet даними з таблиці Factors
da.Fill(ds, "Factors");

// Отримати записи з таблиці Factors
DataRow[] matchRows = ds.Tables["Factors"].Select("GoalFactorReference = 1");

// Пройти по усіх факторах згенерувати перелік
StringBuilder htmlStr = new StringBuilder("");
htmlStr.Append("<ul>");
foreach (DataRow row in matchRows)
{
    htmlStr.Append("<li>");
    htmlStr.Append(row["FactorName"].ToString());
    htmlStr.Append("</li>");
}
htmlStr.Append("</ul>");
HtmlContent.Text = htmlStr.ToString();
```





Прив'язка даних

- **Прив'язка даних** – засіб, що дозволяє асоціювати джерело даних з елементом керування для автоматичного відображення даних у цьому елементі керування
- Виділяють прив'язку даних
  - з одним значенням *(single-value)* – зв'язує властивість елемента керування з джерелом даних, але елемент керування може відображати одне значення
    - » підтримують TextBox, LinkButton, Image, спискові елементи тощо
  - з множиною значень *(repeated-value)* – елементи керування можуть відображати набори значень
    - » підтримують ListBox, GridView тощо

```
DropDownList1.DataSource = ds;
DropDownList1.DataMember = "Student";
DropDownList1.DataTextField = " StudentName";
DropDownList1.DataBind();
```

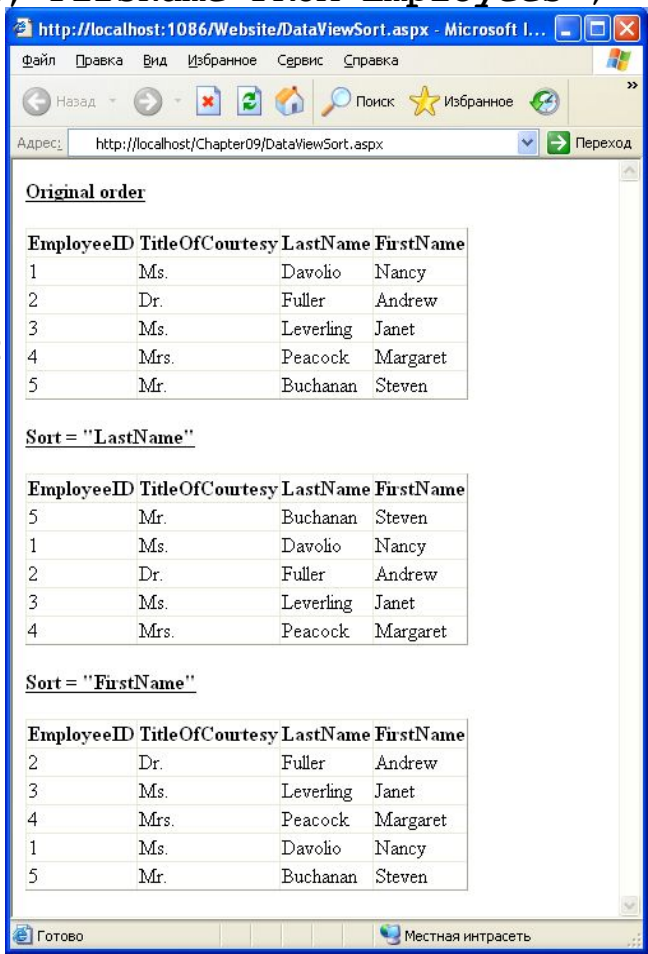
```
GridView1.DataSource = ds;
GridView1.DataMember = "Student";
DropDownList1.DataBind();
```

Клас DataView

- Клас DataView - визначає зовнішнє подання об'єкта DataTable, тобто подання даних у DataTable, яке може включати користувацькі налаштування фільтрації та сортування
  - Дозволяє показати тільки підмножину загального набору даних таблиці , без необхідності обробляти або змінювати дані
  - Кожен об'єкт DataTable має DataView за промовчанням, хоча допускається створювати безліч об'єктів DataView для подання різних видів однієї таблиці
- Властивості
  - Sort – задає один або декілька розділених комою порядків (полів) сортування стовпців
  - RowFilter - задає підмножини рядків на підставі значень їх стовпців

Приклад: сортування за допомогою DataView

```
// Створити Connection, DataAdapter та DataSet
string connectionString =
    WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
String sql =
   "SELECT TOP 5 EmployeeID, TitleOfCourtesy, LastName, FirsName FROM Employees";
SqlDataAdapter da = new SqlDataAdapter(sql, con);
DataSet ds = new DataSet();
// Наповнити DataSet
da.Fill(ds, "Employees");

// Прив'язати оригінальні дані до элемента №1
grid1.DataSource = ds.Tables["Employees"];
// Сортувати за прізвищем та прив'язати до елемента №2
DataView view2 = new DataView(ds.Tables["Employees"]);
view2.Sort = "LastName";
grid2.DataSource = view2;
// Сортувати за іменем та прив'язати до елемента №3
DataView view3 = new DataView(ds.Tables["Employees"]);
view3.Sort = "FirstName";
grid3.DataSource = view3;

// Ініціювати процес прив'язки даних
Page.DataBind();
```

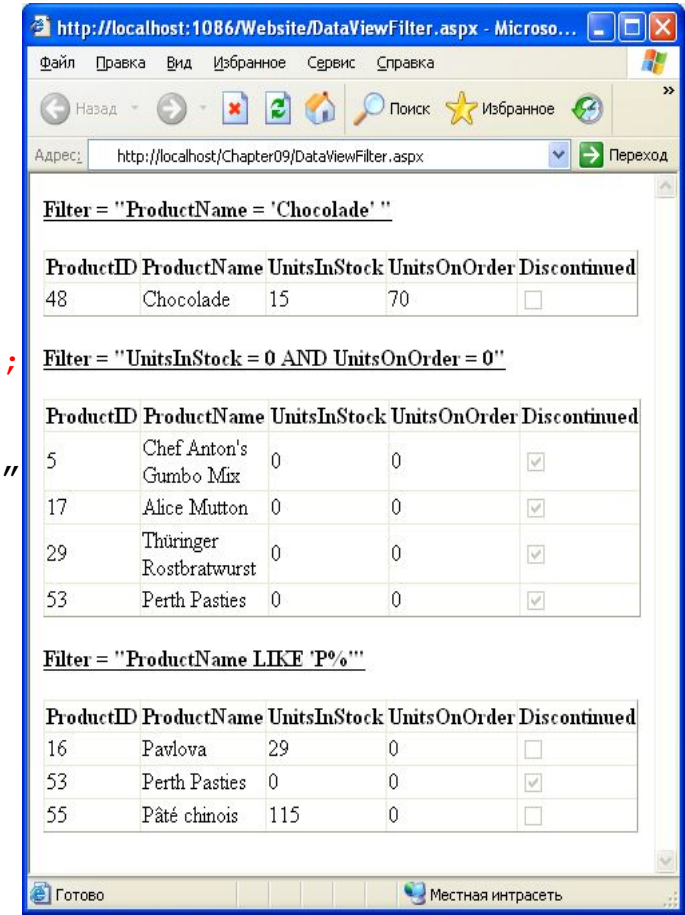


Операції фільтрації

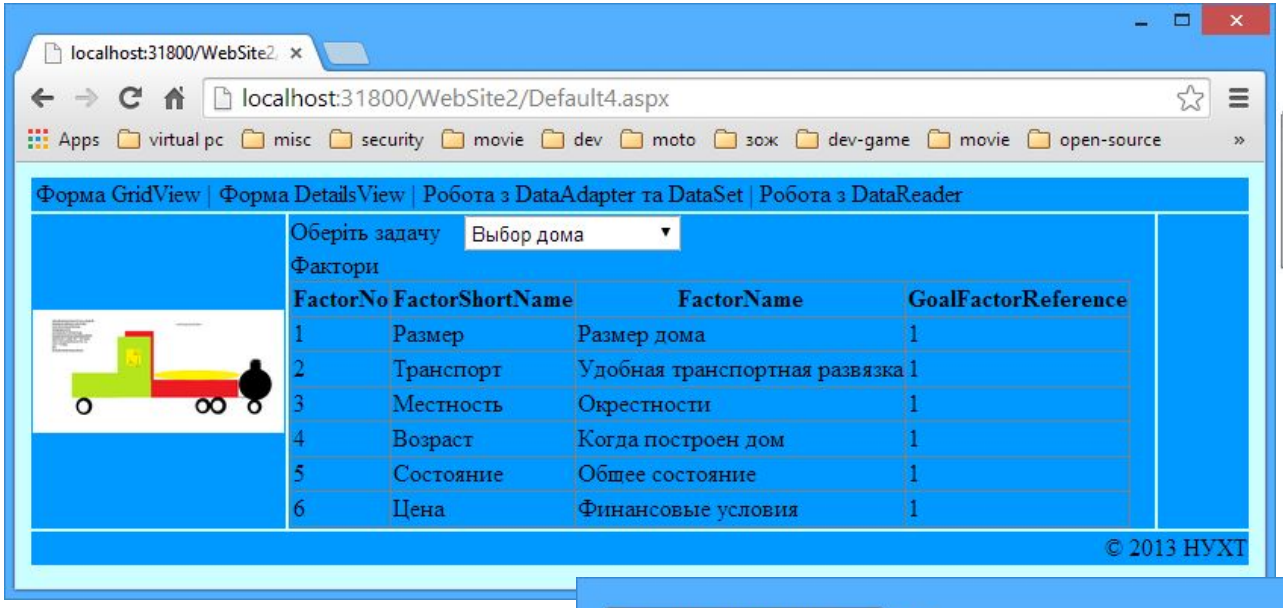
Операція	Опис
<, >, <= та >=	Виконують порівняння більш ніж одного значення
<> та =	Виконують перевірку на еквівалентність
NOT	Звертає на протилежний логічний вираз. Може використовуватися у поєднанні з будь-якої іншої конструкцією
BETWEEN	Вказує діапазон включно. Наприклад, Units BETWEEN 5 AND 15 вибирає рядки, у яких значення стовпця Units знаходиться в межах від 5 до 15
IS NULL	Перевіряє стовпець на null-значення
IN(a, b, c)	Коротка форма операції OR з одним і тим же полем. Перевіряє еквівалентність значення стовпця будь-якого з перерахованих значень (a, b та c)
LIKE	Виконує перевірку відповідності рядкового значення шаблону
+	Складає два числа або "склеює" два рядка
-	Віднімає одне числове значення з іншого
*	Перемножує два числових значення
/	Ділить одне числове значення на інше
%	Обчислює модуль (залишок від ділення одного числа на інше)
AND	Комбінує більше однієї логічної конструкції
OR	Комбінує більше однієї логічної конструкції

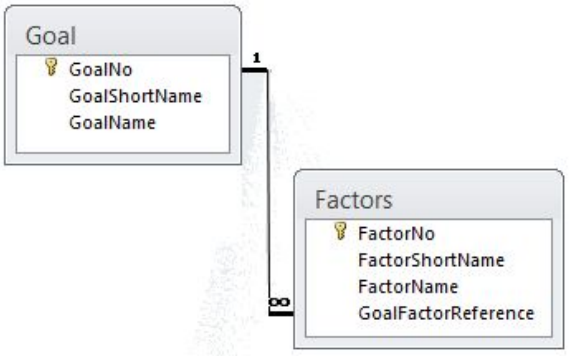
Приклад: фільтрація за допомогою DataView

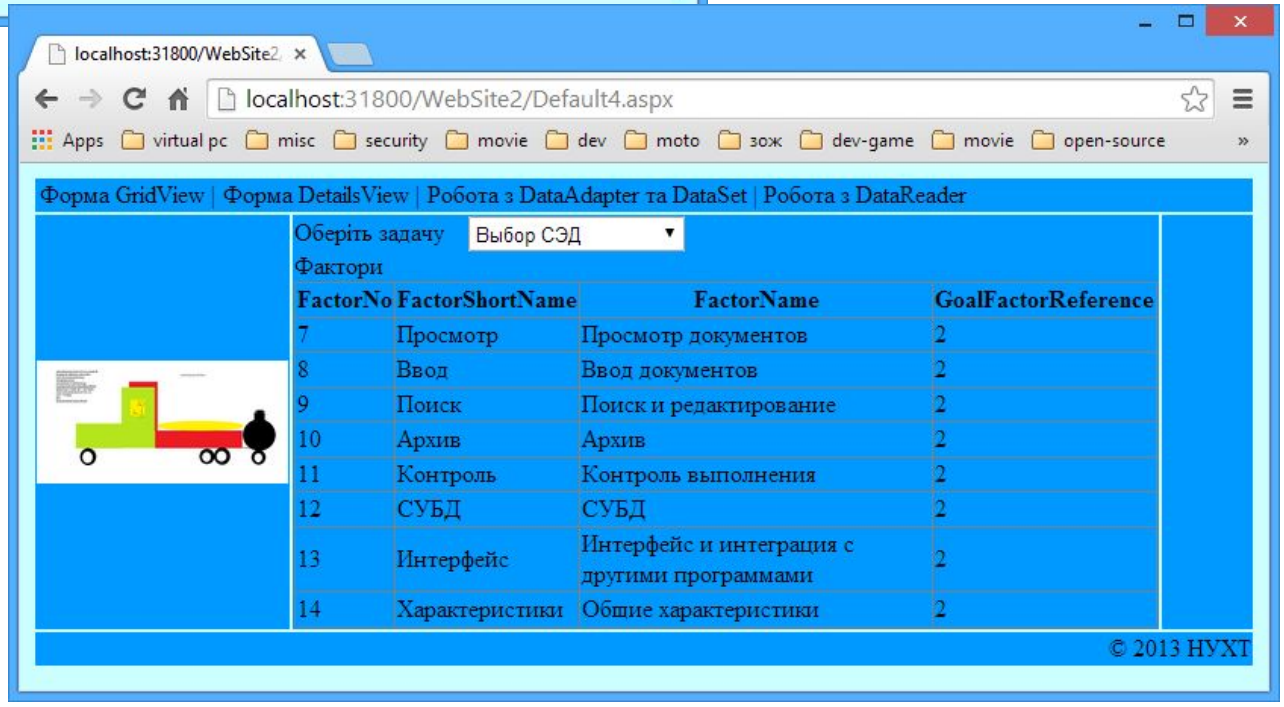
```
string connectionString =
    WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
string sql = "SELECT ProductID, ProductName, UnitsInStock, UnitsOnOrder, " +
    "Discontinued FROM Products";
SqlDataAdapter da = new SqlDataAdapter(sql, con);
DataSet ds = new DataSet();
da.Fill(ds, "Products");

// Фільтрувати продукт Chocolade
DataView view1 = new DataView(ds.Tables["Products"]);
view1.RowFilter = "ProductName = 'Chocolade'";
GridView1.DataSource = view1;

// Фільтрувати продукты, яких немає у замовленнях
// та на складі
DataView view2 = new DataView(ds.Tables["Products"]);
view2.RowFilter = "UnitsInStock = 0 AND UnitsOnOrder = 0";
GridView2.DataSource = view2;

// Фільтрувати продукти, чия назва починається з літери "P"
DataView view3 = new DataView(ds.Tables["Products"]);
view3.RowFilter = "ProductName LIKE 'P%'";
GridView3.DataSource = view3;

this.DataBind();
```



Лабораторна робота №5 (частина 2)







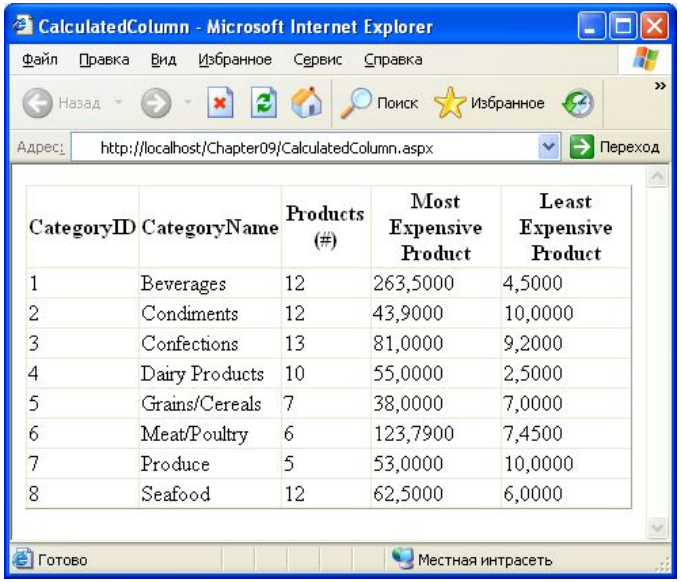
Розширене фільтрування з відношеннями

- DataView дозволяє застосовувати деякі складні вирази фільтрації, напр., на основі відношень
- Фільтруючий вираз є комбінацією
  - відношення, що пов'язує дві таблиці
  - агрегатної функції, напр., AVG(), MAX(), MIN() або COUNT(), що застосовується до даних у пов'язаних записах
- Приклад: показати категорії, що містять продокти, дорожчі за 50 долларів

```
// Визначення відношення між Categories та Products
DataRelation relat = new DataRelation ("CatProds",
    ds.Tables["Categories"].Columns["CategoryID"],
    ds.Tables["Products"].Columns["CategoryID"]);
// Додати відношення до DataSet.
ds.Relations.Add(relat);

// Застосування рядка з умовою до GridView
DataView view1 = new DataView(ds.Tables["Categories"]);
view1.RowFilter = "MAX(Child(CatProds).UnitPrice) > 50";
GridView1.DataSource = view1;
```

Приклад: обчислювані стовпці

```
string connectionString =
    WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString;
SqlConnection con = new SqlConnection(connectionString);
string sqlCat = "SELECT CategoryID, CategoryName FROM Categories";
string sqlProd = "SELECT ProductName, CategoryID, UnitPrice FROM Products";
SqlDataAdapter da = new SqlDataAdapter(sqlCat, con);
DataSet ds = new DataSet();
...
// Визначення відношення між Categories та Products
DataRelation relat = new DataRelation("CatProds",
  ds.Tables["Categories"].Columns["CategoryID"],
  ds.Tables["Products"].Columns["CategoryID"]);
// Додати відношення до DataSet.
ds.Relations.Add(relat);
// Створити обчислювані стовпці
DataColumn count = new DataColumn( "Products (#)",
  typeof(int), "COUNT(Child(CatProds).CategoryID)");
DataColunm max = new DataColumn(
  "Most Expensive Product", typeof(decimal),
  "MAX(Child(CatProds).UnitPrice)");
DataColumn min = new DataColumn( "Least Expensive Product", typeof(decimal),
"MIN(Child(CatProds).UnitPrice)");
// Додати стовбці
ds.Tables["Categories"].Columns.Add(count);
ds.Tables["Categories"].Columns.Add(max);
ds.Tables["Categories"].Columns.Add(min);
// Показати дані
GridView1.DataSource = ds.Tables["Categories"];
GridView1.DataBind();
```



CategoryID	CategoryName	Products (#)	Most Expensive Product	Least Expensive Product
1	Beverages	12	263,5000	4,5000
2	Condiments	12	43,9000	10,0000
3	Confections	13	81,0000	9,2000
4	Dairy Products	10	55,0000	2,5000
5	Grains/Cereals	7	38,0000	7,0000
6	Meat/Poultry	6	123,7900	7,4500
7	Produce	5	53,0000	10,0000
8	Seafood	12	62,5000	6,0000

DataReader vs. DataAdapter

- DataReader допускає швидке та ефективне односпрямоване читання даних
- DataReader менш гнучкий, ніж DataAdapter (не можна редагувати дані, не можна повернутися до прочитаного раніше запису, вимагає монопольного доступу до активного з'єднання)

Крос-платформне програмування

Лекція 11

Розширені елементи керування даними. Основи прив'язки даних

30 квітня, 2014

Примітка: слайди лекції підготовлені за матеріалами http://jskreator.narod.ru/proaspnet20cs/glance.htm

Прив'язка даних

- **Прив'язка даних** – засіб, що дозволяє асоціювати джерело даних з елементом керування для автоматичного відображення даних у цьому елементі керування
- Виділяють прив'язку даних
  - з одним значенням *(single-value)* – зв'язує властивість елемента керування з джерелом даних, але елемент керування може відображати одне значення
    - » підтримують TextBox, LinkButton, Image тощо

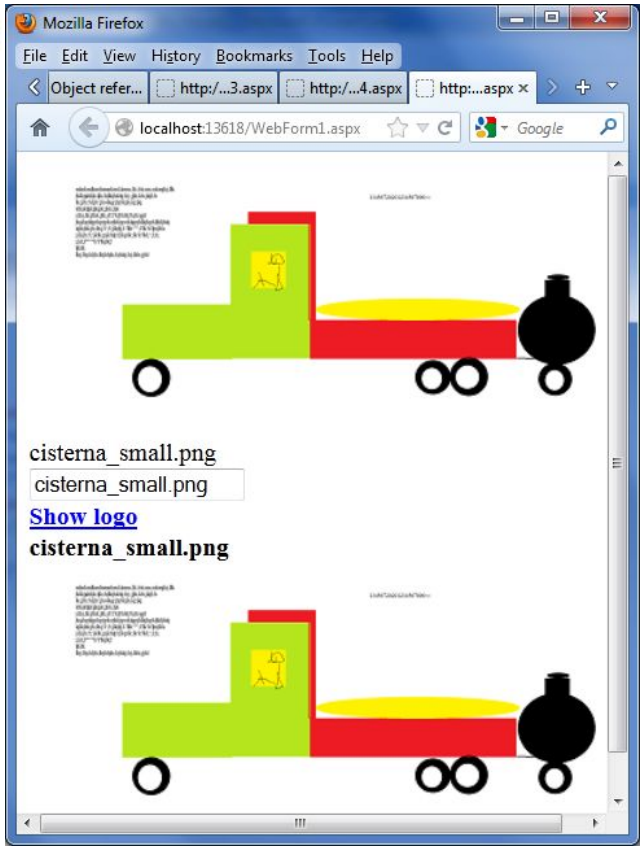
```
<%# EmployeeName%>
<%# GetUserName(ID)%>
<%# 1 + (2 * 20)%>
<%# "Іван" + "Іванов"%>
<%# Request.Browser.Browser%>
```

  - з множиною значень *(repeated-value)* – елементи керування можуть відображати набори значень
    - » підтримують ListBox, GridView тощо

Приклад: прив'язка одного значення

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
<form id="Form1" method="post" runat="server">
<asp:Image ID="Image1" runat="server" ImageUrl ='<%# FilePath %>' /><br />
<asp:Label ID="Label1" runat="server" Text='<%# FilePath %>' /><br />
<asp:TextBox ID="TextBox1" runat="server" Text='<%# GetFilePath() %>' /><br />
<asp:HyperLink ID="HyperLink1" runat="server"
  NavigateUrl='<%# LogoPath.Value %>' Font-Bold="True" Text="Show logo" />
<br />
<input type="hidden" ID="LogoPath" runat="server" value="cisterna_small.pgn">
    <b><%# FilePath %></b>
<br />
<img src='<%# GetFilePath() %>'>
</form>
</body>
</html>
```

```
protected string GetFilePath()
{
    return "cisterna_small.png";
}
protected string FilePath
{
    get { return "cisterna_small.png"; }
}
protected void Page_Load(object sender, EventArgs e)
{
    this.DataBind();
}
```



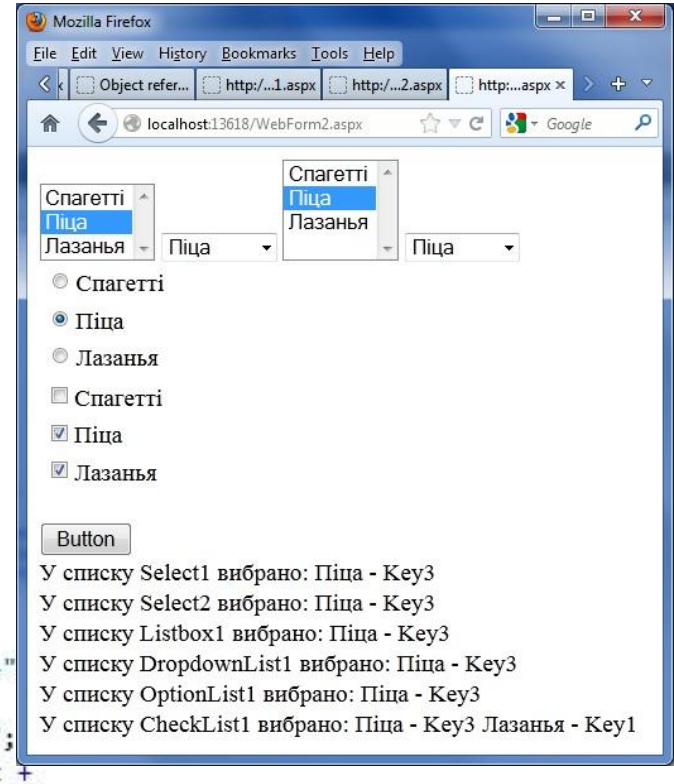
Прив'язка множини значень

- **Спискові ЕК, що підтримують прив'язку множини значень**
  - **усі ЕК, що генерують свій код з використанням дескриптора <select> (HtmlSelect, ListBox та DropDownList тощо)**
  - **прапорці або перемикачі CheckBoxList та RadioButtonList**
  - **список з мітками або пронумерованими пунктами BulletedList**

Властивість	**Опис**
DataSource	Містить відображувані дані, реалізує інтерфейс **ICollection**
DataSourceID	Пов'язує списковий ЕК з елементом-джерелом даних, що згенерує необхідний об'єкт даних автоматично (використовується або **DataSource**, або **DataSourceID**)
DataTextField	Вказує стовпець (у разі рядка таблиці) або властивість (у разі об'єкта) елемента даних, які містять значення, що відображається на сторінці
DataTextFormatString	Специфікує необов'язковий рядок формату, який ЕК буде використовувати для форматування кожного **DataTextValue** перед його відображенням
DataValueField	Зберігає унікальний ідентифікатор або поле первинного ключа, щоб пізніше використати його для вилучення решти даних, коли користувач вибере конкретний елемент

Приклад: прив'язка множини значень

```
protected void Page_Load(object sender, EventArgs e)
{   if (!Page.IsPostBack)
    {   // Створити джерело даних
        Hashtable ht = new Hashtable();
        ht.Add("Лазанья ", "Key1");
        ht.Add("Спагетті", "Key2");
        ht.Add("Піца", "Key3");
        // Встановити властивість DataSource для елемента керування
        Select1.DataSource = ht;
        Select2.DataSource = ht;
        Listbox1.DataSource = ht;
        DropdownList1.DataSource = ht;
        CheckList1.DataSource = ht;
        OptionList1.DataSource = ht;
        Page.DataBind(); // Прив'язати елементи керування
    }
}
protected void Button1_Click(object sender, EventArgs e)
{   LabelResult.Text = "У списку Select1 вибрано: " +
       Select1.Items[Select1.SelectedIndex].Text + " - " + Select1.Value + "<br />"
    LabelResult.Text += "У списку Select2 вибрано: " +
      Select2.Items[Select2.SelectedIndex].Text + " - " + Select2.Value + "<br />";
    LabelResult.Text += "У списку Listbox1 вибрано: " + Listbox1.SelectedItem.Text +
      " - " + Listbox1.SelectedItem.Value + "<br />";
    LabelResult.Text += "У списку DropdownList1 вибрано: " +
      DropdownList1.SelectedItem.Text + " - " + DropdownList1.SelectedItem.Value + "<br />";
    LabelResult.Text += "У списку OptionList1 вибрано: " +
      OptionList1.SelectedItem.Text + " - " + OptionList1.SelectedItem.Value + "<br />";
    LabelResult.Text += "У списку CheckList1 вибрано: ";
    foreach (ListItem li in CheckList1.Items)
    {
        if (li.Selected)
            LabelResult.Text += li.Text + " - " + li.Value + " ";
    }
}
```

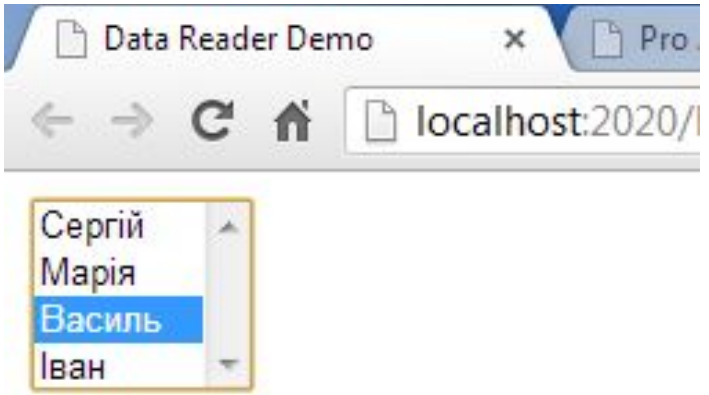


Прив'язка DataReader

- **Класи-джерела даних**
  - **класи колекцій, що повністю містяться у пам'яті, наприклад, Collection, ArrayList, Hashtable та Dictionary**
  - **об'єкт DataReader – заснований на підключенні, односпрямований доступ до БД**
  - **об'єкт DataView – огляд окремого відключеного об'єкта DataTable**
  - **будь-який інший користувацький об'єкт, який реалізує інтерфейс Icollection**

```
<asp:ListBox runat="server" ID="lstNames"
    Size="10" SelectionMode="Multiple"
    DataTextField="FirstName" DataValueField="Id" />
```

```
string connectionString = WebConfigurationManager.ConnectionStrings["kurs"].ConnectionString;
string sql = @"SELECT * FROM Student";

SqlConnection con = new SqlConnection(connectionString);
con.Open();
SqlCommand cmd = new SqlCommand(sql, con);
SqlDataReader reader = cmd.ExecuteReader();
lstNames.DataSource = reader;
lstNames.DataBind();

reader.Close();
con.Close();
```

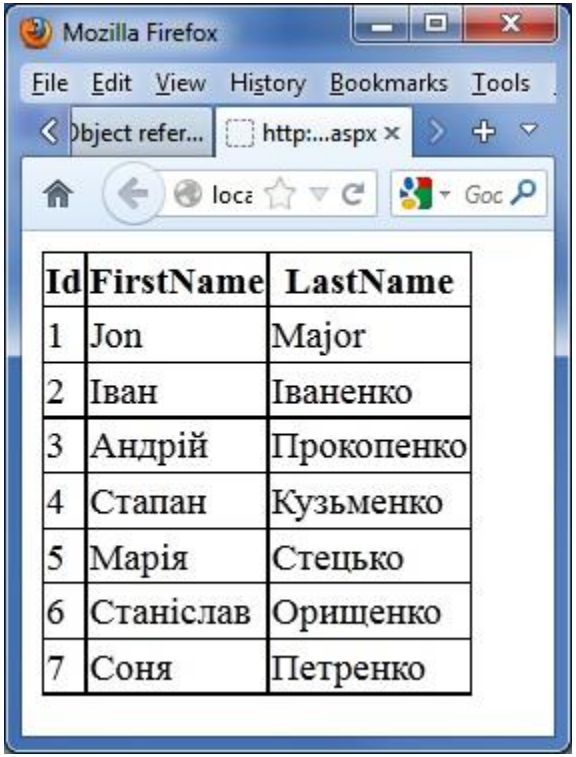


Розширені елементи керування даними

- GridView – табличний елемент керування загального призначення для показу великих таблиць інформації
- DetailsView – показує один запис за раз у таблиці, що має один рядок на поле
- FormView – показує по одному запису за раз, але заснований на шаблонах, що дозволяє комбінувати поля набагато гнучкіше, не обов'язково на основі таблиці
- Menu, TreeView та AdRotator

Приклад: прив'язка до GridView

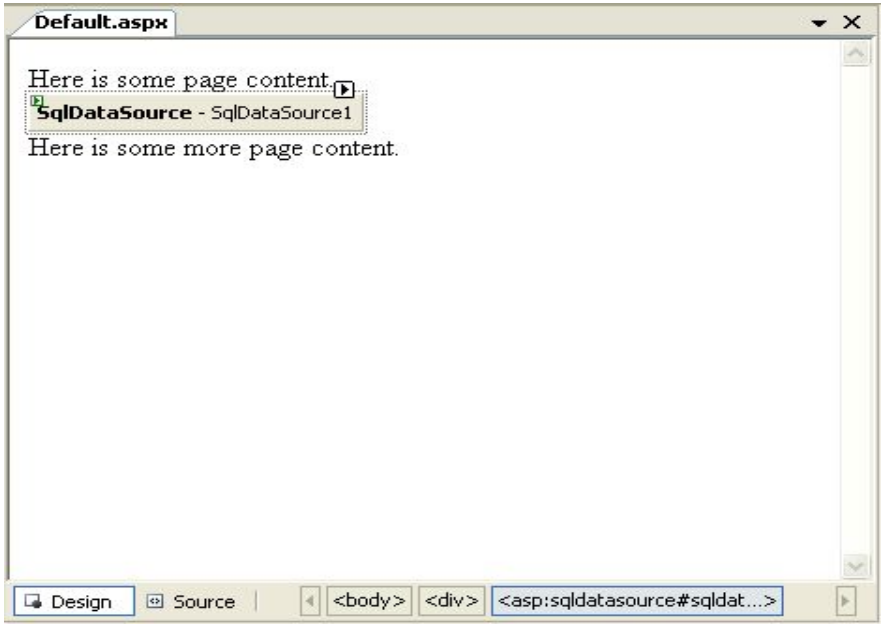
```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        // Створити об'єкти Command і Connection.
        string connectionString = @"data source=.\SQLEXPRESS2008R2;Initial Catalog=kurs;Integrated Security=SSPI;";
        string sql = "SELECT Id, FirstName, LastName FROM Student";
        SqlConnection con = new SqlConnection(connectionString);
        SqlCommand cmd = new SqlCommand(sql, con);
        try
        {   // Відкрити з'єднання і отримати DataReader
            con.Open();
            SqlDataReader reader = cmd.ExecuteReader();

            // Прив'язати DataReader до GridView1
            GridView1.DataSource = reader;
            GridView1.DataBind();

            reader.Close();
        }
        finally
        {   // Закрити з'єднання.
            con.Close();
        }
    }
}
```

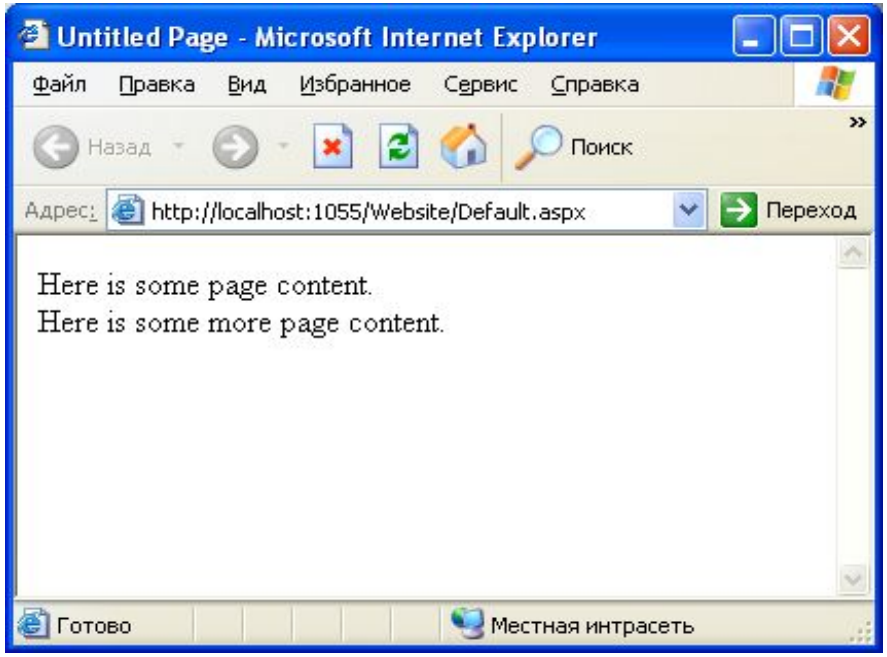


Id	FirstName	LastName
1	Jon	Major
2	Іван	Іваненко
3	Андрій	Прокопенко
4	Стапан	Кузьменко
5	Марія	Стецько
6	Станіслав	Орищенко
7	Соня	Петренко

Елементи керування джерелами даних

- SqlDataSource – дозволяє підключатися до будь-якого джерела даних, яке має постачальника даних ADO.NET
- ObjectDataSource – дозволяє підключатися до користувацького класу доступу до даних
- XmlDataSource – дозволяє підключатися до XML-файлу
- SiteMapDataSource – дозволяє підключатися до файлу Web.Sitemap, що описує навігаційну структуру Web-сайту



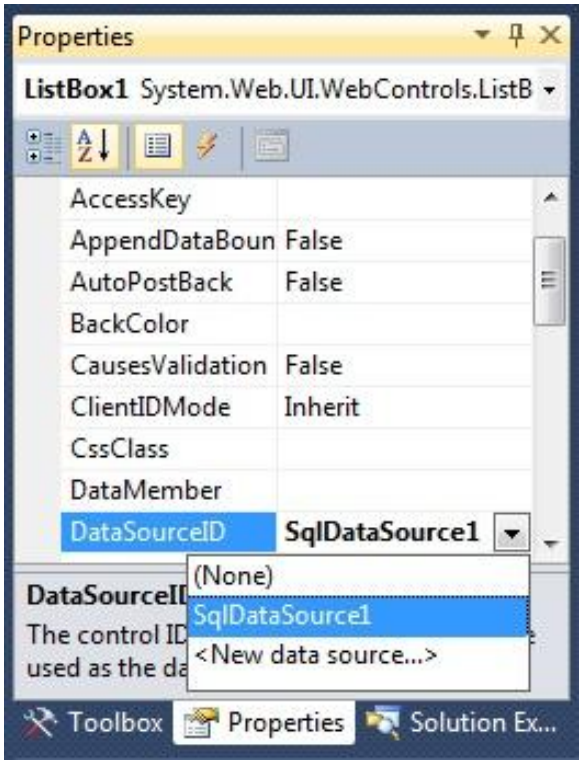


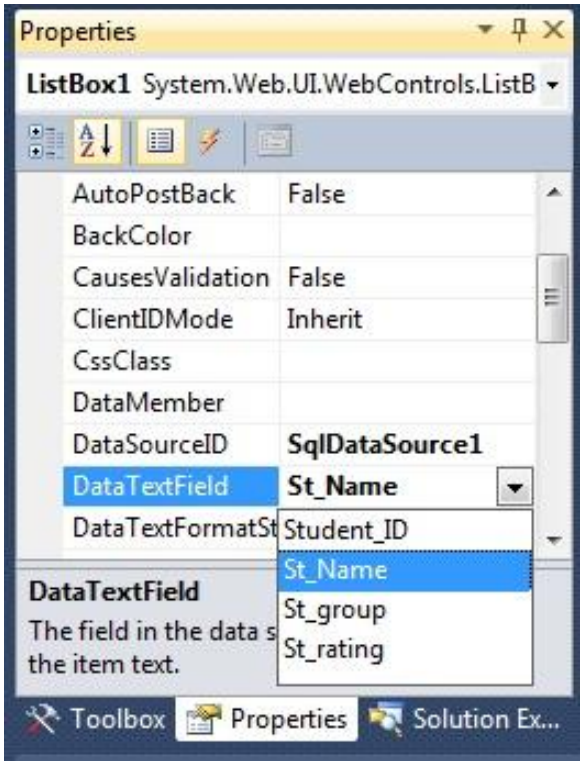
Життєвий цикл сторінки з прив'язкою даних

- **Завдання елементів керування джерелами даних**
  - витягують дані з джерела та застосовують до пов'язаних елементів керування
  - оновлюють джерело даних, коли у пов'язаних елементах керування виконується редагування
- **Порядок виконання прив'язки**
  - Створюється об'єкт сторінки
  - Починається життєвий цикл сторінки, ініціюються події `Page.Init` та `Page.Load`
  - Відбуваються всі інші події
  - Елементи керування джерелами даних виконують будь-які оновлення
  - Збуджується подія `Page.PreRender`
  - Елементи керування джерелами даних виконують необхідні запити і вставляють отримані дані в пов'язані елементів керування
  - Сторінка відображається і розміщується

Прив'язка елементів керування при проектувані

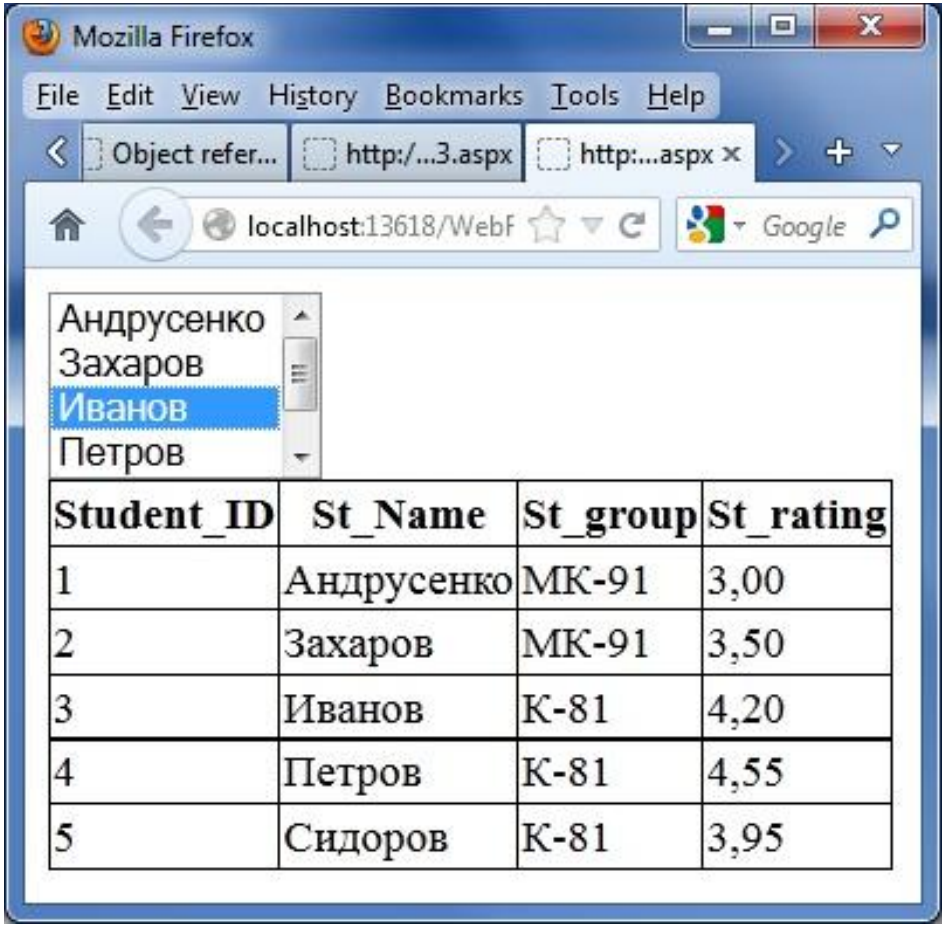
1. Виберіть елемент керування `SqlDataSource` і клацніть на пункті Refresh Schema в інтелектуальному дескрипторі
2. Додайте на форму елемент керування `ListBox`, встановіть властивість `ListBox.DataSourceID`, вибравши джерело даних з випадаючого списку
3. Встановіть `ListBox.DataTextField`, вибравши стовпець `St_Name` з випадаючого списку





Прив'язка елементів керування при проектувані (2)

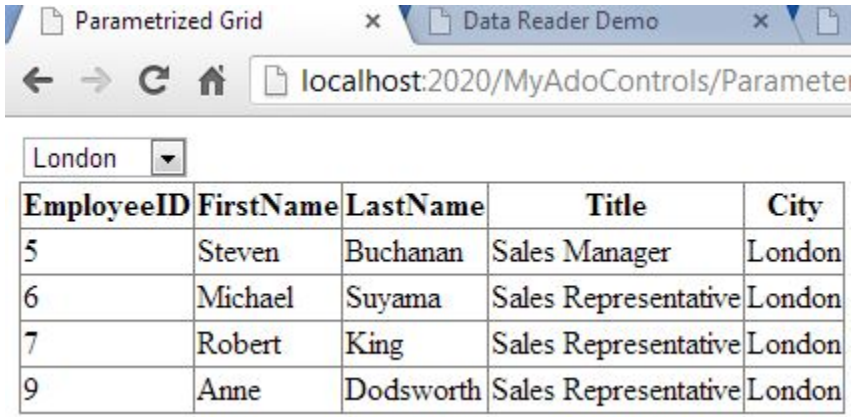
4. Додайте на сторінку `GridView` та встановіть необхідну інформацію про стовпці, оскільки `GridView` може відображати безліч стовпців. У поданні Design одразу відобразяться заголовки стовпців запиту
5. Запустіть сторінку



Student_ID	St_Name	St_group	St_rating
1	Андрусенко	МК-91	3,00
2	Захаров	МК-91	3,50
3	Иванов	К-81	4,20
4	Петров	К-81	4,55
5	Сидоров	К-81	3,95

Приклад: параметризація даних

```
<asp:SqlDataSource ID="sourceEmployeeCities" runat="server"
  ProviderName="System.Data.SqlClient"
  ConnectionString="<%$ ConnectionStrings:Northwind %>"
  SelectCommand="SELECT DISTINCT City FROM Employees">
</asp:SqlDataSource>

<asp:DropDownList ID="lstCities" runat="server"
  DataSourceID="sourceEmployeeCities"
  DataTextField="City" AutoPostBack="True">
</asp:DropDownList>

<asp:SqlDataSource ID="sourceEmployees" runat="server"
  ProviderName="System.Data.SqlClient"
  ConnectionString="<%$ ConnectionStrings:Northwind %>"
  SelectCommand="SELECT EmployeeID, FirstName, LastName,
    Title, City FROM Employees WHERE City=@City">
  <SelectParameters>
   <asp:ControlParameter ControlID="lstCities" Name="City"
     PropertyName="SelectedValue" />
  </SelectParameters>
</asp:SqlDataSource>

<asp:GridView ID="GridView1" AutoGenerateColumns="true" runat="server"
    DataSourceID="sourceEmployees"></asp:GridView>
```

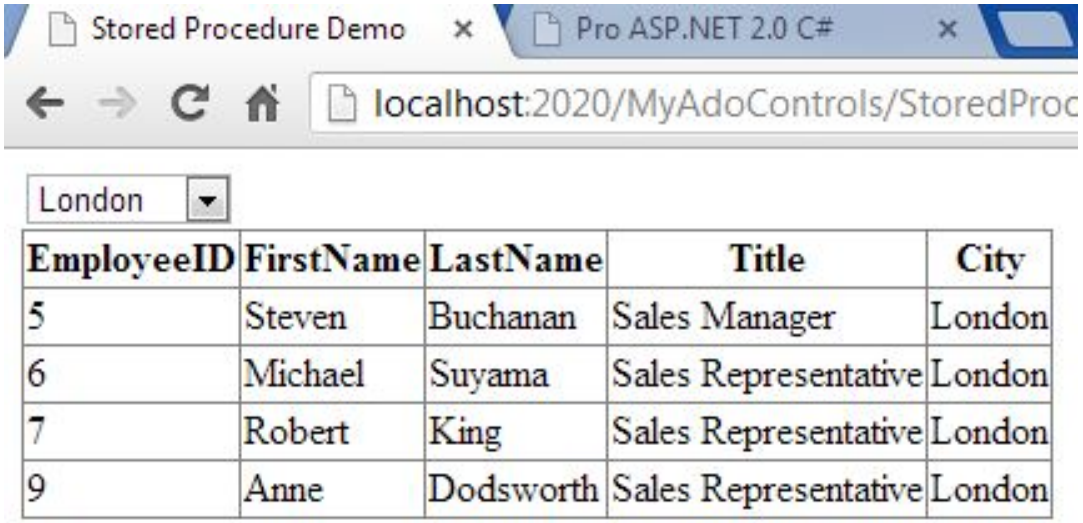


EmployeeID	FirstName	LastName	Title	City
5	Steven	Buchanan	Sales Manager	London
6	Michael	Suyama	Sales Representative	London
7	Robert	King	Sales Representative	London
9	Anne	Dodsworth	Sales Representative	London

Приклад: збережені процедури

```
CREATE PROCEDURE GetEmployeesByCity
  @City varchar(15)
AS
  SELECT EmployeeID, FirstName, LastName, Title, City
  FROM Employees WHERE City=@City
GO
```

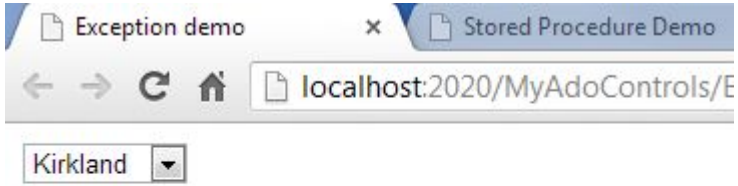
```
<asp:SqlDataSource ID="sourceEmployees" runat="server"
  ProviderName="System.Data.SqlClient"
  ConnectionString="<%$ ConnectionStrings:Northwind %>"
  SelectCommand="GetEmployeesByCity" SelectCommandType="StoredProcedure">
  <SelectParameters>
   <asp:ControlParameter ControlID="lstCities" Name="City"
     PropertyName="SelectedValue" />
  </SelectParameters>
</asp:SqlDataSource>
```



EmployeeID	FirstName	LastName	Title	City
5	Steven	Buchanan	Sales Manager	London
6	Michael	Suyama	Sales Representative	London
7	Robert	King	Sales Representative	London
9	Anne	Dodsworth	Sales Representative	London

Приклад: обробка виключень

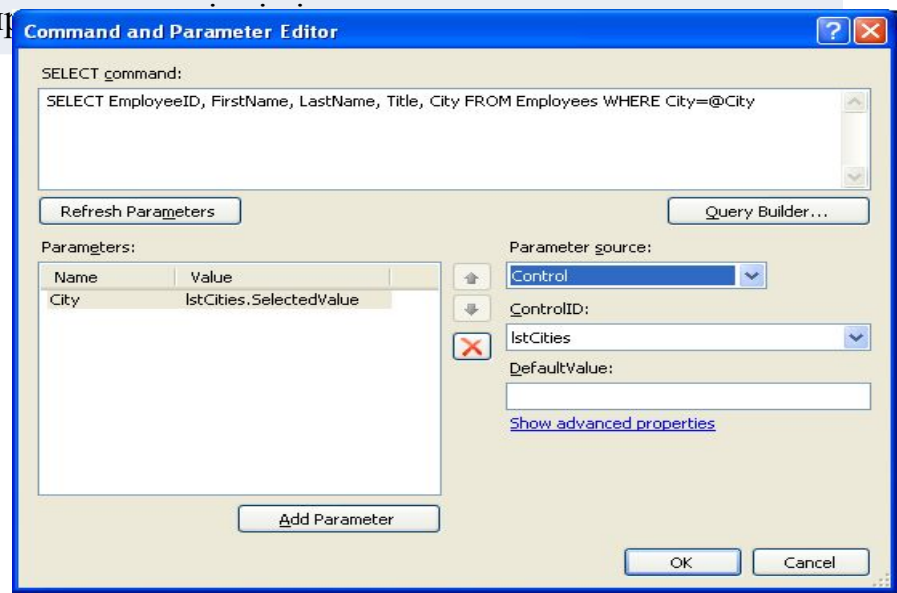
```
<asp:SqlDataSource ID="sourceEmployees" runat="server"
  ProviderName="System.Data.SqlClient"
  ConnectionString="<%$ ConnectionStrings:Northwind %>"
  SelectCommand="SELECT EmployeeID, FirstName, LastName1,
    Title, City FROM Employees WHERE City=@City" OnSelected="sourceEmployees_Selected">
  <SelectParameters>
   <asp:ControlParameter ControlID="lstCities" Name="City"
     PropertyName="SelectedValue" />
  </SelectParameters>
</asp:SqlDataSource>
```




```
protected void sourceEmployees_Selected(object sender,
   SqlDataSourceStatusEventArgs e)
{
    if (e.Exception != null)
    {
        lblError.Text = "При исполнении запроса произошло исключение. ;

        e.ExceptionHandled = true;
    }
}
```

Типи параметрів

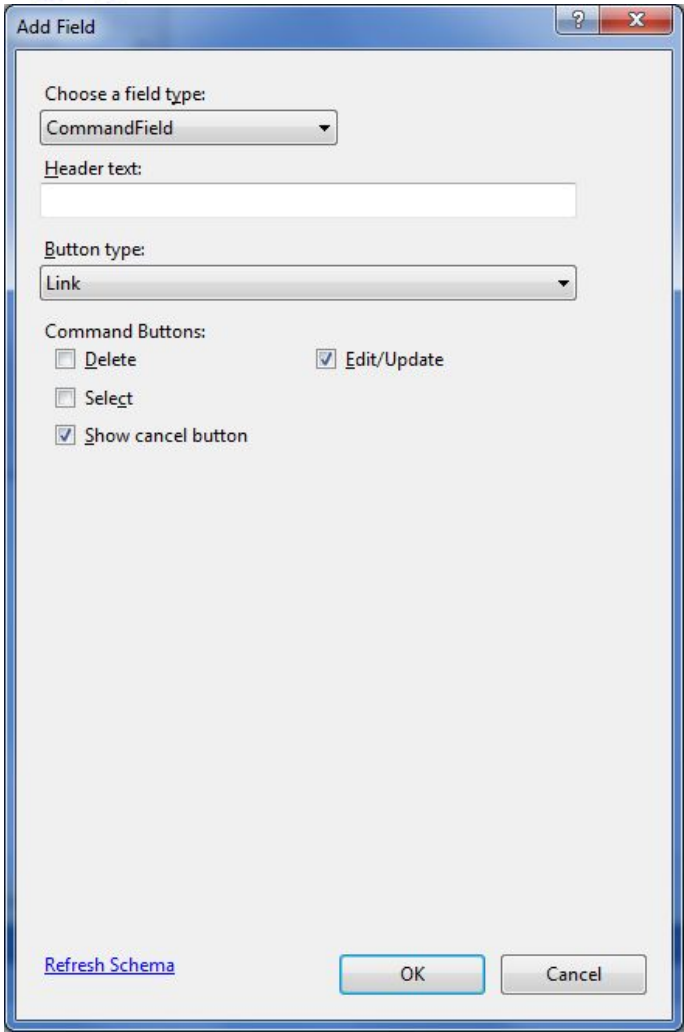
Джерело	Керуючий дескриптор	Опис
Властивість елемента керування	<asp:ControlParameter>	Властивість іншого елемента керування на сторінці.
Рядкове значення запиту	<asp:QueryStringParameter>	Значення з поточного рядка запиту.
Значення стану сеансу	<asp:SessionParameter>	Значення, збережене в поточному сеансі користувача.
Значення cookie-набору	<asp:CookieParameter>	Значення з будь-якого cookie-набору, приєднаного до поточного запиту.
Значення профілю	<asp:ProfileParameter>	Значення з поточного профілю користувача.
Змінна форми	<asp:FormParameter>	Значення, відпр



Оновлення записів

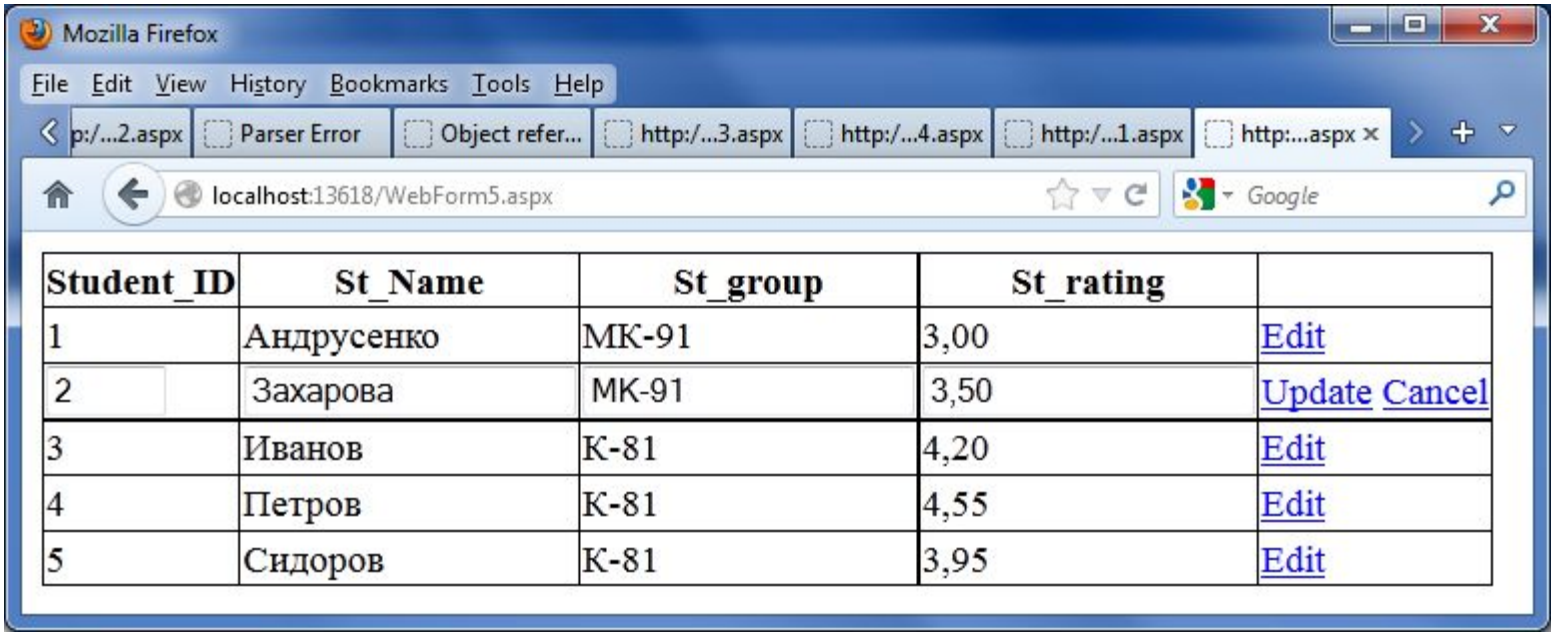
```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:LabDBConnectionString %>"
    SelectCommand="SELECT [Student_ID], [St_Name], [St_group], [St_rating]
      FROM [Student]"
    UpdateCommand="UPDATE Student SET St_Name = @St_Name, St_group = @St_group,
      St_rating = @St_rating WHERE (Student_ID = @Student_ID)">
    <UpdateParameters>
      <asp:Parameter Name="St_Name" />
      <asp:Parameter Name="St_group" />
      <asp:Parameter Name="St_rating" />
      <asp:Parameter Name="Student_ID" />
      </UpdateParameters>
</asp:SqlDataSource>
```

1. Виберіть GridView. В інтелектуальному дескрипторі ЕК виберіть пункт Add New Column
2. У списку Choose a Field Type виберіть CommandField, відзначте прапорці Edit/Update та Show Cancel Button, переконайтеся, що всі інші прапорці не відзначені
3. Клацніть на кнопці OK, щоб додати стовпець з відредагованими ЕК



Оновлення записів (2)

4. У поданні Design одразу відобразиться щойно створений стовпець команд
5. Запустіть сторінку



Student_ID	St_Name	St_group	St_rating	
1	Андрусенко	МК-91	3,00	Edit
2	Захарова	МК-91	3,50	Update Cancel
3	Иванов	К-81	4,20	Edit
4	Петров	К-81	4,55	Edit
5	Сидоров	К-81	3,95	Edit

Недоліки SqlDataSource

- Логіка доступу до даних, вбудовується у сторінку
- Супровід великих застосувань
- Недолік гнучкості
- Незастосовність для інших завдань

Удосконалені елементи керування даними

- Шаблонні елементи керування ASP.NET 1.0
  - DataGrid
  - DataList
  - Repeater
- Нові елементи керування даними ASP.NET 2.0
  - GridView
  - DetailsView
  - FormView

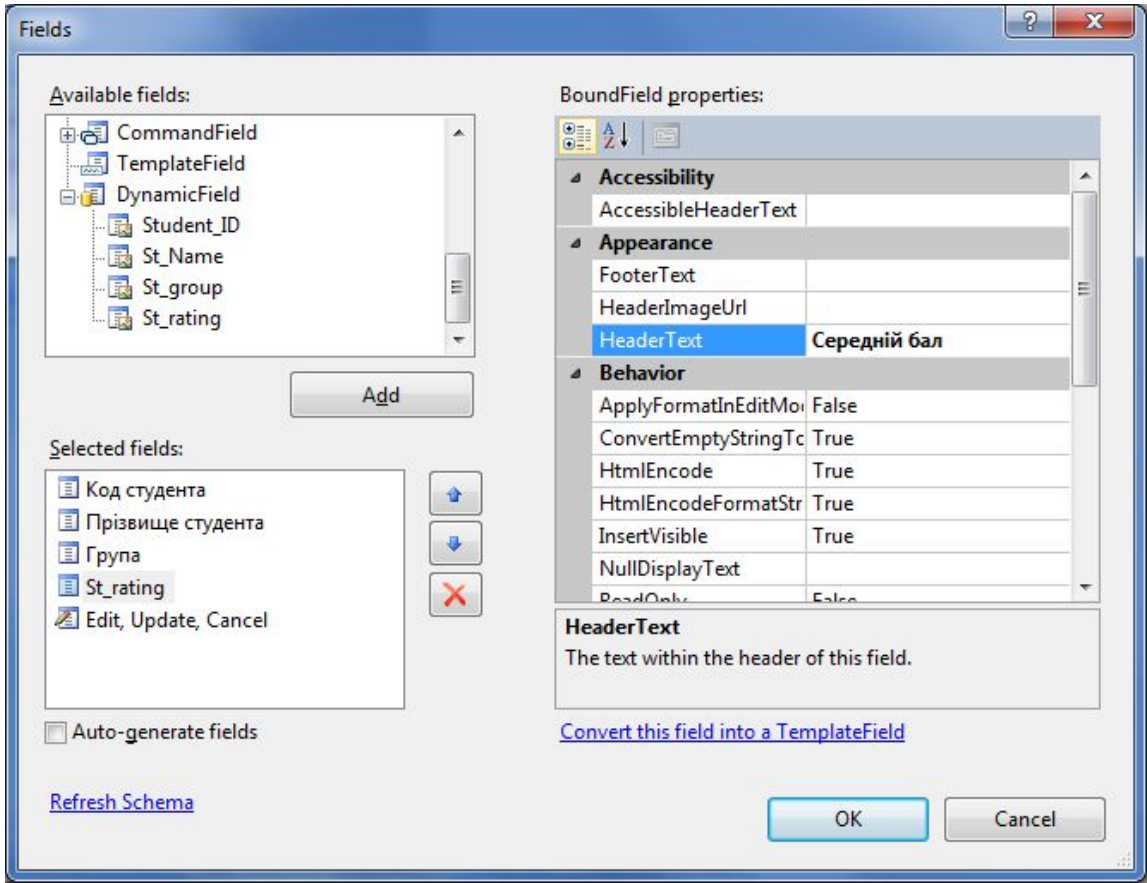
Елемент керування даними GridView

- **GridView – гнучкий табличний елемент керування, призначений для демонстрації даних**
- **Включає широкий діапазон вбудованих засобів, зокрема виділення, розбиття на сторінки і редагування**

Стовпець	Опис
BoundField	Відображає текст поля джерела даних
ButtonField	Відображає кнопку для кожного значення в списку
CheckBoxField	Відображає прапорець для кожного елемента списку. Використовується автоматично для полів, що зберігають значення **true/false**.
CommandField	Надає вибір кнопок редагування.
HyperLinkField	Відображає свій вміст (поле з джерела даних або статичний текст) як гіперпосилання
ImageField	Відображає графічні дані з двійкового поля
TemplateField	Дозволяє специфікувати множинні поля, елементи керування, що налаштовуються, і довільний HTML, використовуючи шаблон

Конфігурування стовпців GridView

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
   DataSourceID="SqlDataSource1">
   <Columns>
     <asp:BoundField DataField="Student_ID" HeaderText="Код студента" />
     <asp:BoundField DataField="St_Name" HeaderText="Прізвище студента" />
     <asp:BoundField DataField="St_group" HeaderText="Група" />
     <asp:BoundField DataField="St_rating" HeaderText="Середній бал" />
     <asp:CommandField ShowEditButton="True" />
   </Columns>
</asp:GridView>
```



Форматування GridView

Форматні рядки зазвичай складаються з заповнювача й індикатора формату, розміщених у фігурних дужках:

{0:C},

де 0 – це значення, яке буде відформатовано,

C – напередвизначений стиль формату.

```
<asp:BoundField DataField = "Price" HeaderText = "Price" DataFormatString = "{0:C}" />
<asp:BoundField DataField="BirthDate" HeaderText="Birth Date"
      DataFormatString="{0:MM/dd/yy}" />
```

Тип	Форматний рядок	Приклад
Грошовий	{0:C}	$1,234.50 Дужки означають від'ємне значення: ($1,234.50). Символ валюти залежить від локальних налаштувань: (?1,234.50)
Науковий (експоненційний)	{0:E}	1.234.50E+004
Процентний	{0:P}	45.6%
Фіксований десятковий	{0:F?}	Залежить від кількості десяткових розрядів після крапки: {0:F3} дасть 123.400, а {0:F0} – 123.
Короткая дата	{0:d}	M/d/yyyy (наприклад, 10/30/2005)

Стилі GridView

Кожен стиль надає об'єкт Style, що включає властивості для:

- вибору кольорів (ForeColor та BackColor);
- додавання рамок (BorderColor, BorderStyle та BorderWidth);
- розмірів рядки (Height та Width);
- вирівнювання рядка (HorizontalAlign та VerticalAlign);
- конфігурування зовнішнього вигляду тексту (Font та Wrap).

Додавання атрибутів стилю:

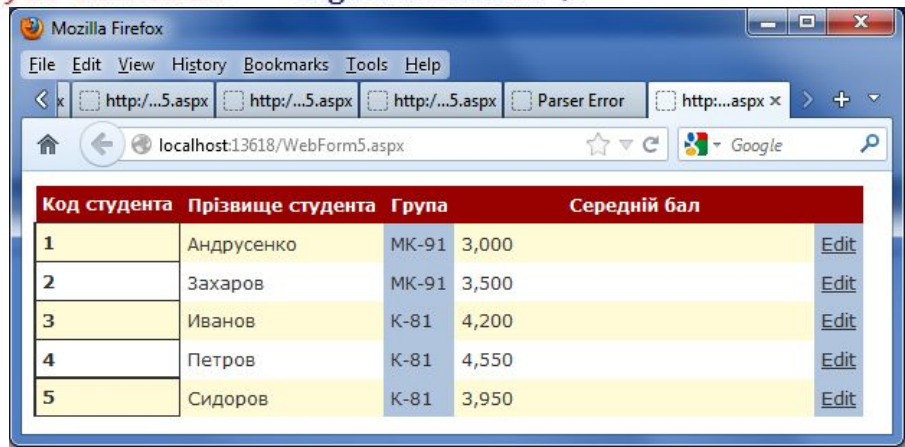
```
<asp:GridView ID="GridView1" runat="server" ... >
    <Columns>
        <asp:BoundField DataField="Student_ID" HeaderText="Код студента"
            ItemStyle-BackColor ="LightSteelBlue" />
    </Columns>
</asp:GridView>
```

Застосування вкладених дескрипторів всередині дискриптора стовпця:

```
<asp:GridView ID="GridView1" runat="server" ... >
    <Columns>
        <asp:BoundField DataField="Student_ID" HeaderText="Код студента"
            SortExpression="Student_ID" >
            <ItemStyle Font-Bold="True" BorderWidth="1" />
        </asp:BoundField>
    </Columns>
</asp:GridView>
```

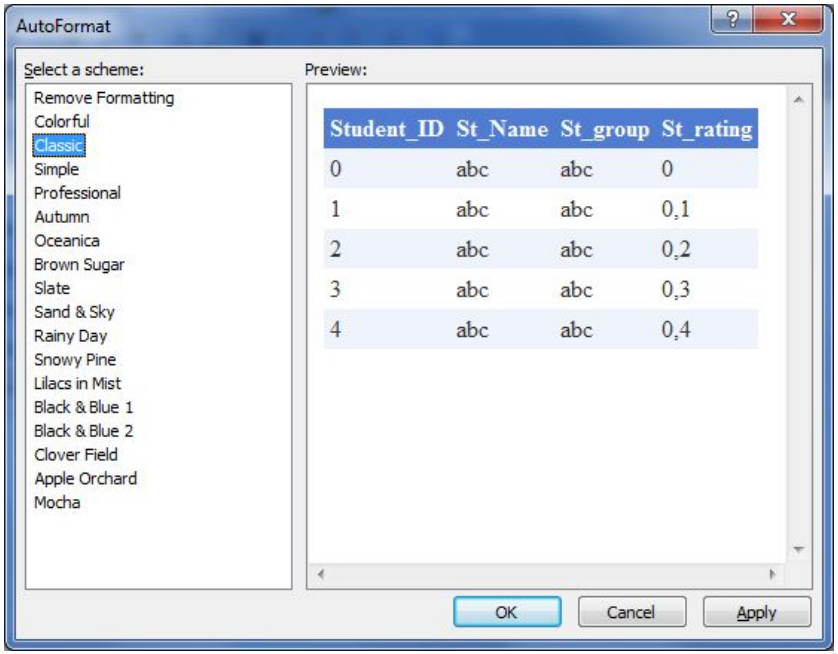
Визначення стилів

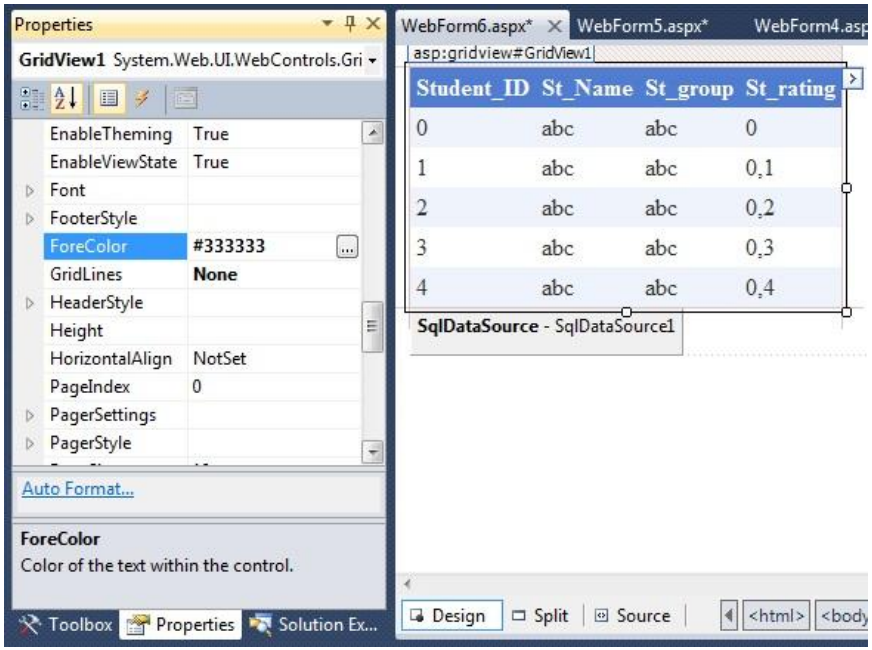
```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    Font-Names = "Verdana" Font-Size = "X-Small" ForeColor = "#333333"
    CellPadding = "4" GridLines = "None" DataSourceID="SqlDataSource1" >
    <HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="White" />
    <RowStyle BackColor="#FFFBD6" ForeColor="#333333" />
    <AlternatingRowStyle BackColor="White" />
    <Columns>
        <asp:BoundField DataField="Student_ID" HeaderText="Код студента"
            SortExpression="Student_ID" >
            <ItemStyle Font-Bold="True" BorderWidth="1" />
        </asp:BoundField>
        <asp:BoundField DataField="St_Name" HeaderText="Прізвище студента"
            SortExpression="St_Name" />
        <asp:BoundField DataField="St_group" HeaderText="Група"
            SortExpression="St_group" >
            <ItemStyle BackColor="LightSteelBlue" />
        </asp:BoundField>
        <asp:BoundField DataField="St_rating" HeaderText="Середній бал"
            SortExpression="St_rating" DataFormatString = "{0:F3}" >
            <ItemStyle Wrap="True" Width="200"/>
        </asp:BoundField>
        <asp:CommandField ShowEditButton="True" ItemStyle-BackColor ="LightSteelBlue"/>
    </Columns>
</asp:GridView>
```



Код студента	Прізвище студента	Група	Середній бал	
1	Андрусенко	МК-91	3,000	Edit
2	Захаров	МК-91	3,500	Edit
3	Иванов	К-81	4,200	Edit
4	Петров	К-81	4,550	Edit
5	Сидоров	К-81	3,950	Edit

Налаштування стилів






```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
    CellPadding="4" DataSourceID="SqlDataSource1" ForeColor="#333333"
    GridLines="None">
    <AlternatingRowStyle BackColor="White" />
    <Columns>
        ...
    </Columns>
    <EditRowStyle BackColor="#2461BF" />
    <FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
    <HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
    <PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />
    <RowStyle BackColor="#EFF3FB" />
    <SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />
    <SortedAscendingCellStyle BackColor="#F5F7FB" />
    <SortedAscendingHeaderStyle BackColor="#6D95E1" />
    <SortedDescendingCellStyle BackColor="#E9EBEF" />
    <SortedDescendingHeaderStyle BackColor="#4870BE" />
</asp:GridView>
```

Перелік додаткових посилань

- *Мэтью Мак-Дональд, Марио Шпушта.* Microsoft ASP.NET 3.5 с примерами на C# 2008 и Silverlight 2 для профес-сионалов.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2009. – 1408 с. - http://jskreator.narod.ru/proaspnet20cs/glance.htm

- *Рихтер Дж.* CLR via C#. Програм-мирование на платформе Microsoft .NET Framework 4.0 на языке C#. 3-е изд. – СПб.: Питер, 2012. – 928 с.
- Корисні ресурси
  - http://msdn.microsoft.com/ru-ru/library/

Дякую за увагу

Модифікація та оновлення

- Структура даних повинна бути відома на період компіляції
- Об'єкт DataSet можна редагувати на клієнтській машині (редагувати записи, додавати або видаляти DataRow)
- Усі виконані у DataSet зміни потраплять до БД після застосування методу DataAdapter.Update
- RowState – стан рядка даних Unchanged/Modified/Added/Deleted/Detached/