CHAPTER 16

4) The Data Controls

When it comes to data binding, not all ASP.NET controls are created equal. In the previous chapter, you saw how data binding can help you automatically insert single values and lists into all kinds of common controls. In this chapter, you'll concentrate on three more advanced controls—GridView, DetailsView, and FormView—that allow you to bind entire tables of data.

The rich data controls are quite a bit different from the simple list controls. For one thing, they are designed exclusively for data binding. They also have the ability to display more than one field at a time, often in a table-based layout or according to what you've defined. They also support higher-level features such as selecting, editing, and sorting.

The rich data controls include the following:

- *GridView*: The GridView is an all-purpose grid control for showing large tables of information. The GridView is the heavyweight of ASP.NET data controls.
- *DetailsView*: The DetailsView is ideal for showing a single record at a time, in a table that has one row per field. The DetailsView also supports editing.
- *FormView*: Like the DetailsView, the FormView shows a single record at a time and supports editing. The difference is that the FormView is based on templates, which allow you to combine fields in a flexible layout that doesn't need to be table based.

Grid view features:

→ Improved datasource bindingcapabilities
→ Tabular rendering - displays data as a table
→ Built - in sorting capability
→ Built - in select, edit and delete capabilities
→ Built - in paging capability
→ Built - in row selection capability
→ multiple key fields
→ programmatic access to the GridView object model to dynamically set properties, handle events and so on
→ Richer design-time capabilities
→ Control over Alternate item, Header, Footer, colors ~~and Data list control~~, Font borders, and soon
→ Slow performance as compared to Repeater and Data list control

GridView Behavior properties:-

Propert	Description
AllowPaging	

Table 16-6. *Paging Members of the GridView*

Property	Description
AllowPaging	Enables or disables the paging of the bound records. It is false by default.
PageSize	Gets or sets the number of items to display on a single page of the grid. The default value is 10.
PageIndex	Gets or sets the zero-based index of the currently displayed page, if paging is enabled.
PagerSettings	Provides a PagerSettings object that wraps a variety of formatting options for the pager controls. These options determine where the paging controls are shown and what text or images they contain. You can set these properties to fine-tune the appearance of the pager controls, or you can use the defaults.
PagerStyle	Provides a style object you can use to configure fonts, colors, and text alignment for the paging controls.
PageIndexChanging and PageIndexChanged events	Occur when one of the page selection elements is clicked, just before the PageIndex is changed (PageIndexChanging) and just after (PageIndexChanged).

Details View control:

drawback: In Pager navigation, whole set of records are retrived from database even though one record data is displayed in the control.

Features of DetailsView control:

→ 1) Tabular rendering, 2) Supports column layout, by default 2 columns at time
3) optional support for Paging and Navigation

Details View Properties:

→ AutoGenerateEditButton
a) Built-in support for edit, insert and delete capabilites

→ AutoGenerateInsertButton

→ Auto GenerateRows

→ BackImageUrl

→ DataMember

→ DataSourceID.

The FormView

If you need the ultimate flexibility of templates, the FormView provides a template-only control for displaying and editing a single record.

The beauty of the FormView template model is that it matches quite closely the model of the TemplateField in the GridView. This means you can work with the following templates:

- ItemTemplate
- EditItemTemplate
- InsertItemTemplate
- FooterTemplate
- HeaderTemplate
- EmptyDataTemplate
- PagerTemplate

Data Source controls:-

→ ASP.NET includes data source controls that allow you to work with different types of data sources such as a database, an XML file, or a middle-tier business object. Data source controls connect to and retrive data from a data source and make it available for other control to bind to, without requiring code

→ you can use each sqlDataSource control you create to retrieve a single query. optinally you can also add corresponding commands for deleting, inserting, and updating rows. .NET Framework contains or includes the following data sources:

Data source control.	Description
sqlDataSource	This data source allows you to connect to any data source that has an ADO.NET data provider. This includes SQL server, Oracle, and OLEDB or ODBC data sources

5)

AccessDataSource	This data source allows you to read and write the data in an Access database file (.mdb)
ObjectDataSource	This data source allows you to connect to a custom data access class.
XmlDataSource	This data source allows you to connect to an XML file.
SiteMapDataSource	This data source allows you to connect to a .sitemap file that describes the navigational structure of your website
EntityDataSource	This data source allows you to query a database by using the LINQ to Entities feature
LinqDataSource	→ This data source allows you to query a database by using the LINQ to SQL feature, which is similar (but somewhat less powerful) predecessor to LINQ to Entities

→ SelectCommand, InsertCommand, DeleteCommand can be added to SqlDataSource to specify the SQL statements

6) LINQ

↳ Language integrated Query

→ Query for retrieving data can be done using different languages:

SQL → for relation data

XQuery → for XML

→ LINQ was introduced in 2008 with .NET framework version 3.5

→ Using LINQ you can select, filter, sort, group, and transform data.

DisAdv of traditional query language:-

1. developers have had to learn a new query language for each type of data source or data format that they must support.

LINQ advantages:-

1. LINQ simplifies this situation by offering a consistent model for working with data across various kinds of data sources and formats.

2. In a LINQ query, you are always working with objects. You use the same basic coding patterns to query and transform data in XML documents, SQL databases, ADO.NET Datasets,

LINQ Providers available with .NET 4.5:

→ LINQ to objects → Every objects
→ LINQ to Dataset → accessing dataset
→ LINQ to XML → " XML
→ LINQ to Entities → EF models

LINQ QUERY OPERATIONS.

1. Filtering operators
2. Join "
3. Projection "
4. Sorting "
5. Grouping "
6. Conversions "
7. Aggregation " → Aggregate, Average, count, condition, count, max, min, sum
8. Quantifier " → All, Any, Contains
9. Partition "
10. Generation "
11. Set "
12. Equality "
13. Element "

- Projections

1. The ability to transform the data you're querying into results with a different structure is called Projection

→ Projection is an operation in which an object is transformed into an altogether new form with only specific properties

Eg:-1

```
var matches = from employee in employees
select employee.FirstName + " " + employee.
                                    LastName;
```

(or)

```
select new { First = employee.FirstName, Last =
                                    employee.LastNam
                                    e };
```

projection operators :- select, SelectMany

operator:-

select → The operator projects values on basis of a transform function

SelectMany →

Concatenation: does not remove duplications, Concat

Join operators →

Join → Join two sequences on basis of matching keys

GroupJoin → Join two sequences and group the matching elements

Grouping operators

group--by-- or group-- into

IEnumeration collection of type IGrouping<key, elements>	To.Lookup → Execute a grouping operation in which a sequence of key pairs are returned.

operator sorting

orderBy → The operator sort values in acending order

orderByDescending → descending order

ThenBy →

ThenBy descending →

Reverse →

objects of type Employee

Eg:-

```
var matches = from employee in employees
    orderby employee.LastName, employee.FirstName
    select employee;
```

Filtering:-

* It is an operation to restrict the result set such that it has only selected elements satisfying a particular condition.

operators

where → Filter values based on a predicate function

OfType →

Conversion operators :- ASEnumerable, ASQueryable

Cast, OfType, ToArray, ToDictionary, ToList, ToLookup

LINQ

ToLook operator:

```
namespace ConsoleApplication1
{
    class Employee
```

LINQ EX

```
using system;
using system.Linq;
  //find all elements <5
class LINQ-EX2
  {
  static void main()
      {
   int[]numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0};

var lowNums =
          from n in numbers
        where n<5
          select n;
  Console.writeLine("Numbers < 5:");
   for each (var x in lowNums)
   {
    console.writeLine(x);
   }
  }
}
```