ASP.NET programming

Data Source controls
Data Binding
LINQ to SQL

Data Source controls (1)

- LinqDataSource Control
  - Declarative markup can be used to write all the conditions that are required in order retrieve, filter, order, and group the data
  - Can be configured to handle update, insert, and delete operations automatically
  - Requires less code than other data source controls.
- SqlDataSource Control
  - Retrieves and modifies data using SQL commands.
  - Works with Microsoft SQL Server, OLE DB, ODBC, and Oracle databases
- EntityDataSource Control
  - Supports data binding scenarios based on the Entity Data Model (EDM).
- ObjectDataSource Control
  - Works with a business object or other class in Web applications that rely on middle-tier business objects to manage data.
  - Interacts with an object that implements one or more methods to retrieve or modify data.
    - The methods must return a DataSet, DataTable, or DataView object, or an object that implements the IEnumerable interface.

Data Source controls (2)

- XmlDataSource Control
  - Reads and writes XML data so that one can work with it using controls such as the TreeView and Menu controls.
  - Can read either an XML file or string of XML.
  - Can use the schema to expose data using typed members.
  - XSLT transformation as well as XPath expressions can be applied to the XML data.
- AccessDataSource Control
  - is a specialized version of the SqlDataSource control, designed to work specifically with Microsoft Access .mdb files.
- SiteMapDataSource Control
  - Works with ASP.NET site maps and provides site navigation data.
  - Most commonly used with the Menu control.

Data binding

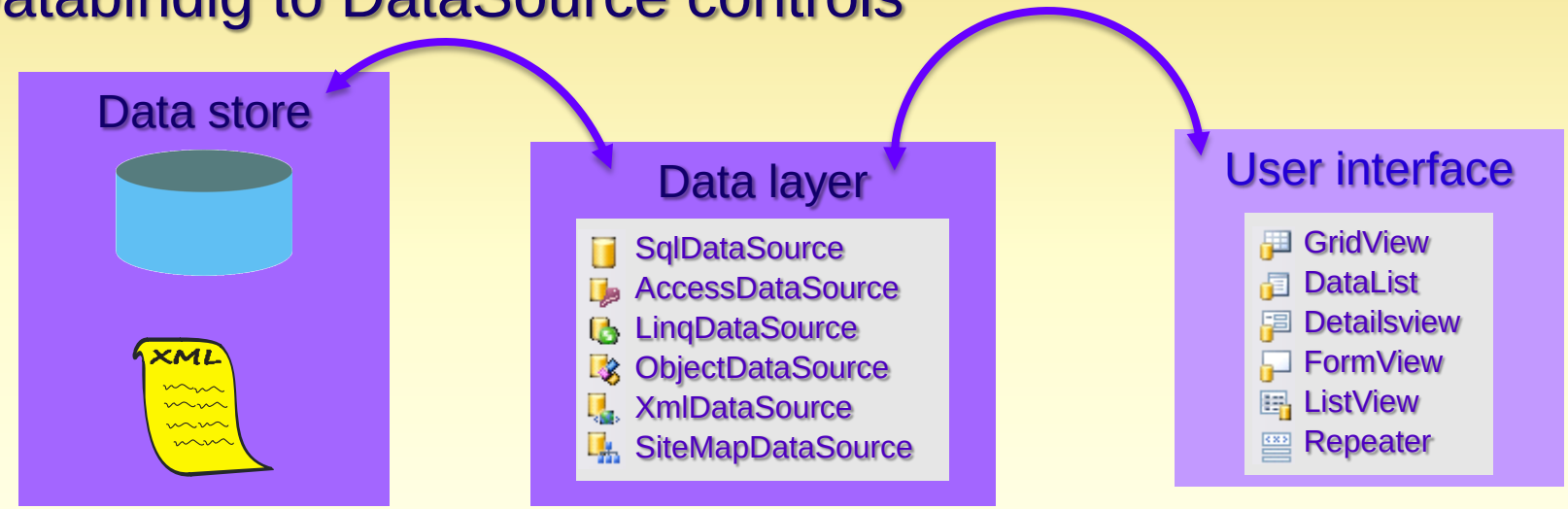
Almost any control can be bound to any datasource...

```
protected void Page_Load(object sender, EventArgs e) {
  string[] s ={"ASP.NET", "ADO.NET", "C# (CSharp)", "VB.NET"};
  lstTopics.DataSource = s;
  lstTopics.DataBind();
}
```

ASP.NET
ADO.NET
C# (CSharp)
VB.NET

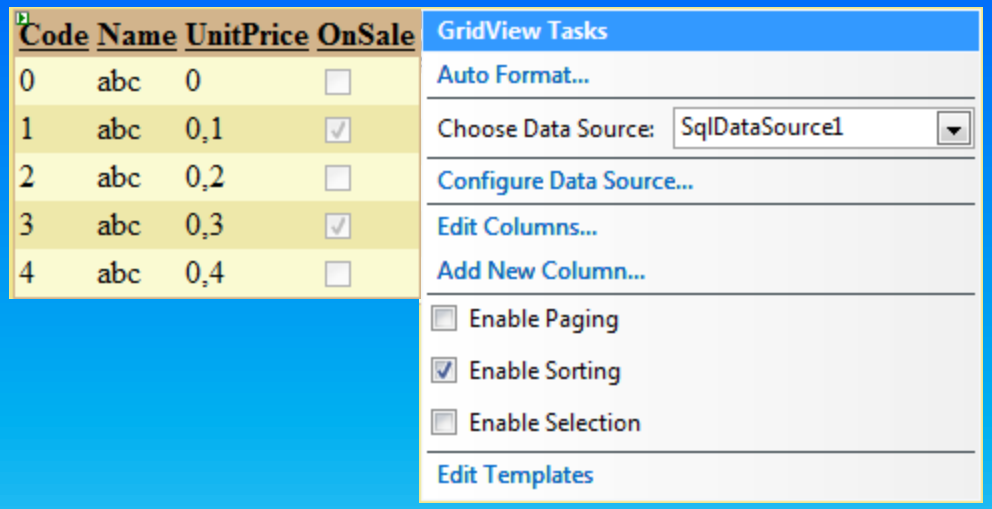
...but the real benefit comes with the data controls

Databindig to DataSource controls

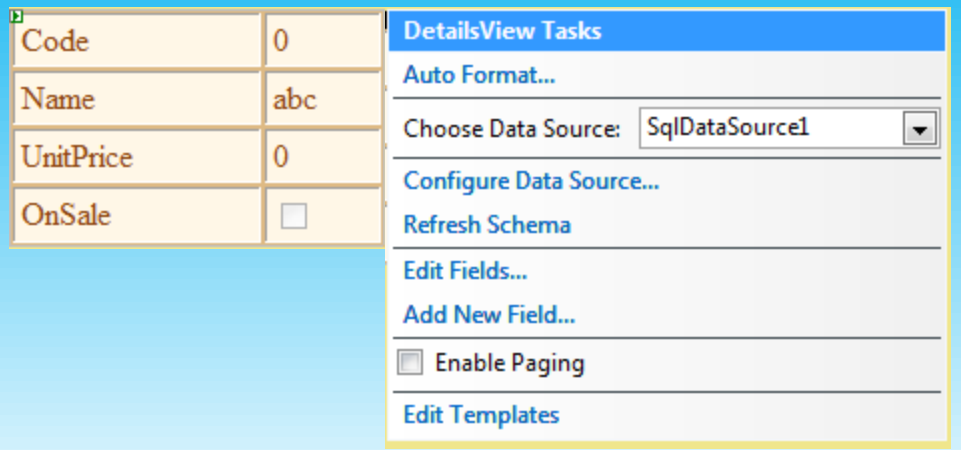


GridView and DetailView controls

- GridView
  - Displays data as a table
  - Sorting, paging
  - Editing, deleting



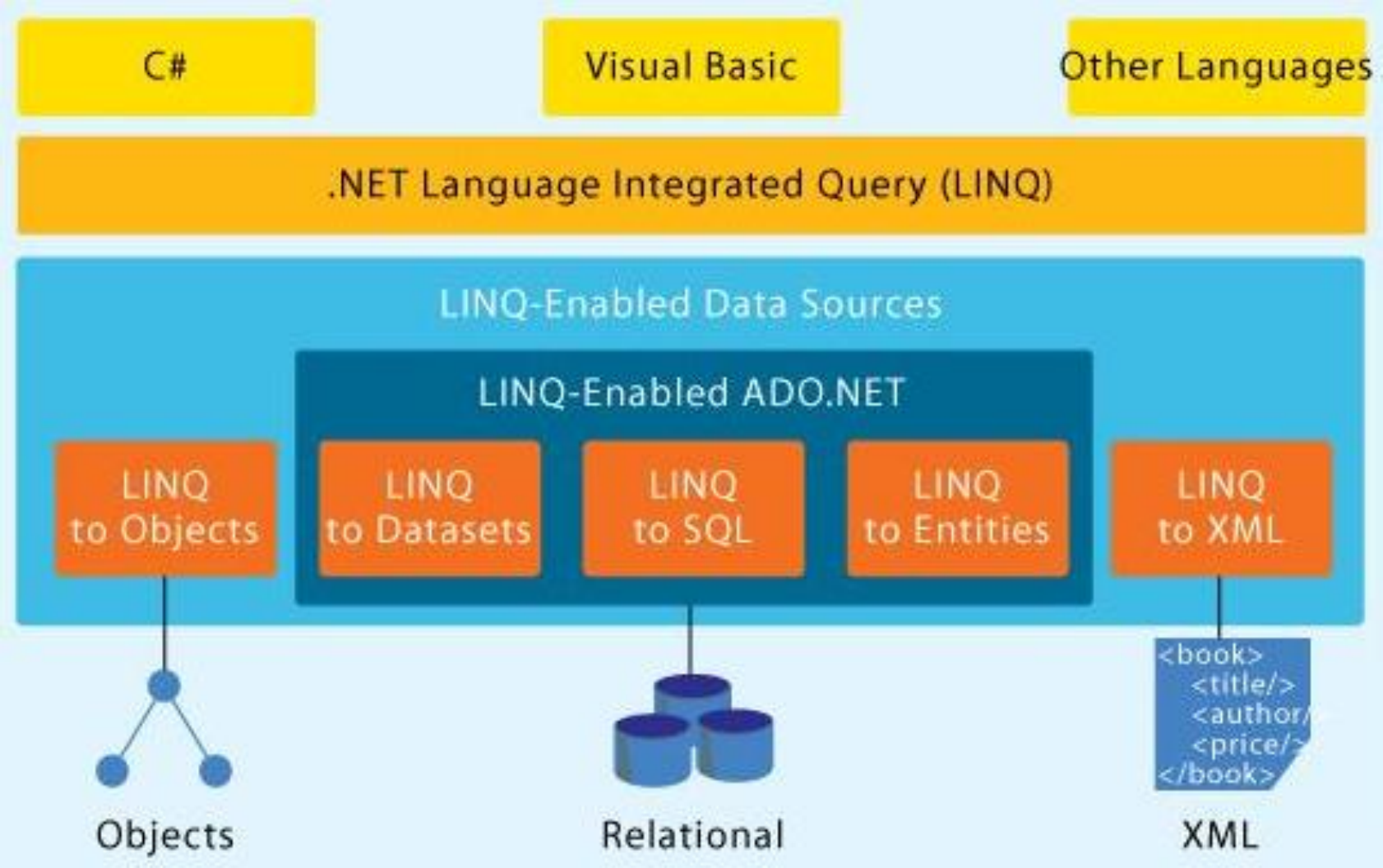
- DetailsView
  - Renders a single record at a time
  - Paging through multiple records
  - Inserting, editing, deleting



DEMO

GridView and DetailsView controls
Data binding to EntityDataSource

LINQ Architecture



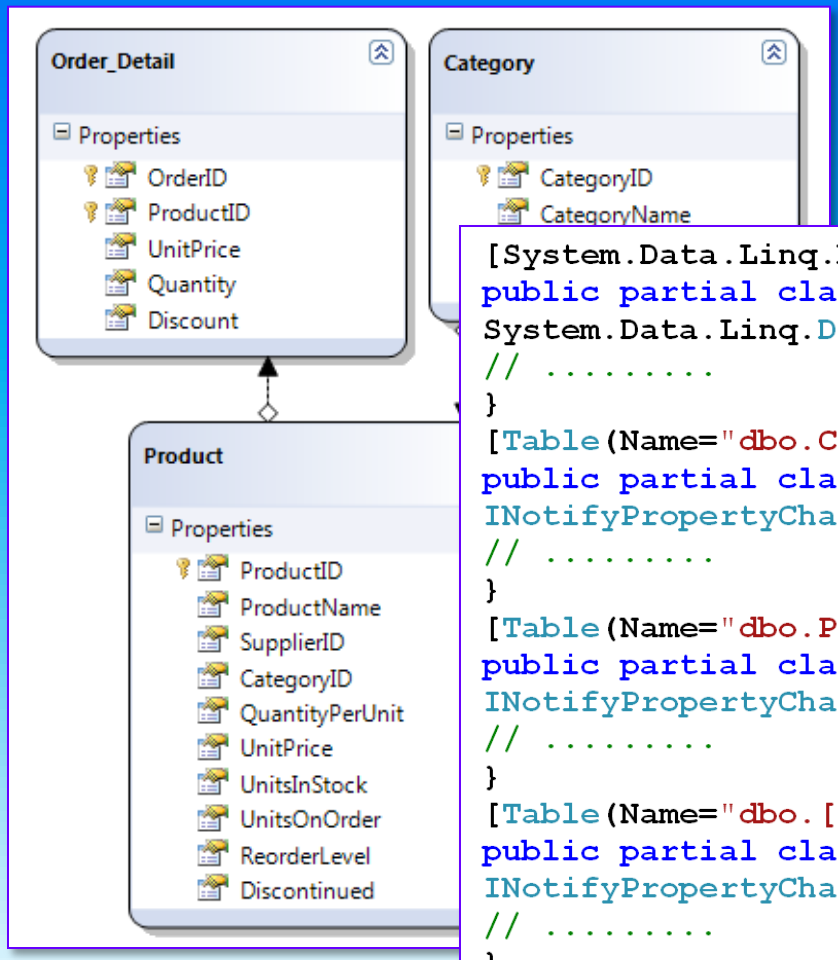
LINQ to SQL
Overview

- Is an ORM (Object Relational Mapping) implementation
  - Allows a relational database to be modelled using .NET classes
  - Visual Studio provides an easy way of visualization and modelling the database as a LINQ to SQL object model
- Allows the database queries to be programmed against the strogly typed object model
  - SIDUing the data (select, insert, delete, update) can be implemented
  - Error detecting at compile time
  - Visul Studio intellisense is available
- Supports transactions, views, and stored procedures
- Provides an easy way to integrate data validation and business logic rules into the data model
- Generates Transact-SQL commands to be executed

LINQ to SQL

Object Relational Mapping in Visual Studio

Model visualization



Model implementation

```
[System.Data.Linq.Mapping.DatabaseAttribute(Name="NORTHWIND")]
public partial class NorthwindDataContext :
System.Data.Linq.DataContext {
// .........
}
[Table(Name="dbo.Categories")]
public partial class Category :
INotifyPropertyChanging, InotifyPropertyChanged {
// .........
}
[Table(Name="dbo.Products")]
public partial class Product :
INotifyPropertyChanging, InotifyPropertyChanged {
// .........
}
[Table(Name="dbo.[Order Details]")]
public partial class Order_Detail :
INotifyPropertyChanging, InotifyPropertyChanged {
// .........
}
```

LINQ to SQL
Code Examples (1)

Query syntax

```
NorthwindDataContext db = new NorthwindDataContext();
IEnumerable<Product> products =
    from p in db.Products
    where p.Category.Products.Count > 5
    select p;
```

Same query with method syntax (using Lambda expression)

```
NorthwindDataContext db = new NorthwindDataContext();
IEnumerable<Product> products =
    db.Products.Where(p => p.Category.Products.Count > 5);
```

LINQ to SQL
Directly Execute SQL Queries

- As long as the column names in the tabular results match column properties of your entity class, LINQ to SQL creates objects out of any SQL query
  - The ExecuteQuery method also allows parameters.

```
Northwind db = new Northwind(@"c:\northwind.mdf");
IEnumerable<Customer> results =
  db.ExecuteQuery<Customer>
  ("SELECT contactname FROM customers WHERE city = {0}",
   "London");
```

LINQ to SQL
Standard Query Operators

- Declared in
  - System.Collection.Generic.IEnumerable<T>
  - System.Linq.IQueryable<T>

- Defined as static methods in
  - System.Linq.Enumerable
  - System.Linq. Queryable

- Restriction operator
  - Where
- Projection operators
  - Select
  - SelectMany
- Partitioning operators
  - Take
  - Skip
  - TakeWhile
  - SkipWhile
- Join operators
  - Join
  - GroupJoin
- Concatenation operator
  - Concat
- Ordering operators
  - OrderBy/ThenBy
  - Reverse
- Grouping operators
  - GroupBy
- Set operators
  - Distinct
  - Union
  - Intersect
  - Except
- Conversion operators
  - ToSequence
  - ToArray
  - ToList
  - ToDictionary
  - ToLookup
  - OfType
  - Cast
- Equality operator
  - EqualAll
- Element operators
  - First
  - FirstOrDefault
  - Last
  - LastOrDefault
  - Single
  - SingleOrDefault
  - ElementAt
  - ElementAtOrDefault
  - DefaultIfEmpty
- Generation operators
  - Range
  - Repeat
  - Empty
  - Quantifiers
  - Any
  - All
  - Contains
- Aggregate operators
  - Count
  - LongCount
  - Sum
  - Min
  - Max
  - Average
  - Aggregate

DEMO

LINQ query

LINQ to SQL
Code Examples (2)

Query products from the database (query syntax)

```
NorthwindDataContext db = new NorthwindDataContext();
var query = from p in db.Products
            where p.Category.CategoryName == "Beverages"
            select p;
```

Update a product in the database (method syntax)

```
NorthwindDataContext db = new NorthwindDataContext();
Product product =
  db.Products.Single(p => p.ProductName == "Chang");
product.UnitPrice = 55;
db.SubmitChanges();
```

Delete a product from the database (query syntax)

```
NorthwindDataContext db = new NorthwindDataContext();
var discontinuedProducts = from p in db.Products
                           where p.Discontinued == true
                           select p;
db.Product.DeleteAllOnSubmit(discontinuedProducts);
db.SubmitChanges();
```

LINQ to SQL
Code Examples (3)

Insert a new category and two new products into the database

```
NorthwindDataContext db = new NorthwindDataContext();

// Create new Category and Products
Category category = new Category();
category.CategoryName = "Toys";
Product product1 = new Product ();
Product product2 = new Product ();

// Associate Product with Category
category.Products.Add(product1);
category.Products.Add(product2);

// Add Category to database and save changes
db.Categories.InsertOnSubmit(category);
db.SubmitChanges();
```

DEMO

SIDU operations using LINQ to SQL

LINQ to SQL

Summary

- Easy way to implement database applications using either query syntax or method syntax, both integrated in programming language
- Interactive way of creating the ORM classes needed
- Full IntelliSense support in design time
- Strong type checking in compile time
- Exception handling available in full extend
- Easy way to integrate data validation and business logic rules into the data model by using partial methods

Links

LINQ to SQL reference

Lambda expressions