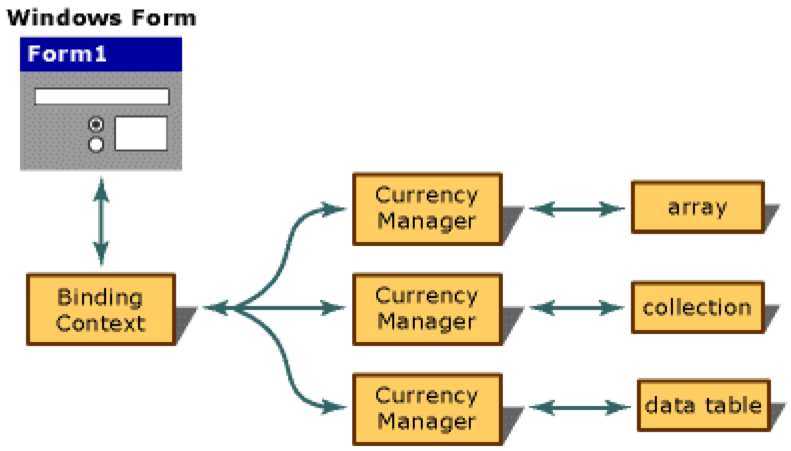
Slide's de spus in plus:

SLIDE 3

Data binding provides a way for developers to create a read/write link between the controls on a form and the data in their application (their data model). Classically, data binding was used within applications to take advantage of data stored in databases. Windows Forms data binding allows you to access **data from databases** as well as data in other structures, such as **arrays** and **collections**.



Binding Context
Each Windows Form has at least one **BindingContext** object that manages the **CurrencyManager** objects for the form. **For each data source on a Windows Form, there is a single CurrencyManager object.** Because there may be multiple data sources associated with a Windows Form, the BindingContext object enables you to retrieve any particular CurrencyManager object associated with a data source.

CurrencyManager
The CurrencyManager is used to **keep data-bound controls synchronized with each other (showing data from the same record).** The CurrencyManager object does this by managing a collection of the bound data supplied by a data source. For each data source associated with a Windows Form, the form maintains at least one CurrencyManager. Because there may be more than one data source associated with a form, the BindingContext object manages all of the CurrencyManager objects for any particular form. More broadly, all container controls have at least one BindingContext object to manage their CurrencyManagers.
An **important property** of the CurrencyManager is the **Position** property. *Currency* is a term used to refer to the currentness of position within a data structure. You can use the Position property of the CurrencyManager class to determine the current position of all controls bound to the same CurrencyManager.
For example, imagine a collection consisting of two columns called "ContactName" and "Phone". Two TextBox controls are bound to the same data source. When the Position property of the common CurrencyManager is set to the fourth position within that list (corresponding to the fifth name, because it is zero-based), both controls display the appropriate values (the fifth "ContactName" and the fifth "Phone") for that position in the data source.

<u>Data Binding and Windows Forms</u>

In Windows Forms, you can bind traditional data sources and any structure that contains data. You can bind to an array of values that you calculate at run time, read from a file, or derive from the values of other controls. You can bind any property of any control to the data source. In traditional data binding, you typically bind the display property (for example, the Text property of a TextBox control) to the data source. With the .NET Framework, you also have the option of setting other properties through binding as well. You might use binding to perform the following tasks:

- Setting the graphic of an image control.
- Setting the background color of one or more controls.
- Setting the size of controls.

Essentially, data binding is an automatic way of setting any run-time accessible property of any control on a form.

Types of Data Binding: simple binding and complex binding (The ability of a control to bind to more than one data element, typically more than one record in a database. Complex binding is also called list-based binding. Examples of controls that support complex binding are the DataGridView, ListBox, and ComboBox controls.)

BindingSource Component

To simplify data binding, Windows Forms enables you to bind a data source to the BindingSource component and then bind controls to the BindingSource. You can use the BindingSource in simple or complex binding scenarios. In either case, the BindingSource acts as an intermediary between the data source and bound controls providing change notification currency management and other services.

SLIDE 10

Adding DataRelations (ADO.NET)

In a DataSet with multiple DataTable objects, you can use DataRelation objects to relate one table to another, to navigate through the tables, and to return child or parent rows from a related table. The arguments required to create a DataRelation are a name for the DataRelation being created, and an array of one or more DataColumn references to the columns that serve as the parent and child columns in the relationship. After you have created a DataRelation, you can use it to navigate between tables and to retrieve values.

Adding aDataRelationto aDataSetadds, by default, aUniqueConstraintto the parent table and a ForeignKeyConstraint to the child table.
Example: creates a DataRelation using two DataTable objects in a DataSet. Each DataTable contains a column named CustID, which serves as a link between the two DataTable objects. The example adds a single DataRelation to the Relations collection of the DataSet. The first argument in the example specifies the name of the DataRelation being created. The second argument sets the parent DataColumn and the third argument sets the child DataColumn.

```
customerOrders.Relations.Add("CustOrders",
customerOrders.Tables["Customers"].Columns["CustID"],
customerOrders.Tables["Orders"].Columns["CustID"]);
```

A DataRelation also has a Nested property which, when set to true, causes the rows from the child table to be nested within the associated row from the parent table when written as XML elements using WriteXml.

Navigating DataRelations

A DataRelation allow navigation from one DataTable to another within a DataSet. So, you can retrieve all the related DataRow objects in one DataTable when given a single DataRow from a related DataTable. For example, after establishing a DataRelation between a table of customers and a table of orders, you can retrieve all the order rows for a particular customer row using GetChildRows. The following code example creates a DataRelation between the Customers table and the Orderstable of a DataSet and returns all the orders for each customer.

Navigating DataRelations

A DataRelation allow navigation from one DataTable to another within a DataSet. So, you can retrieve all the related DataRow objects in one DataTable when given a single DataRow from a related DataTable. For example, after establishing a DataRelation between a table of customers and a table of orders, you can retrieve all the order rows for a particular customer row using GetChildRows. The following code example creates a DataRelation between the Customers table and the Orderstable of a DataSet and returns all the orders for each customer.

```
DataRelation customerOrdersRelation = customerOrders.Relations.Add("CustOrders",
customerOrders.Tables["Customers"].Columns["CustomerID"],
customerOrders.Tables["Orders"].Columns["CustomerID"]);

foreach (DataRow custRow in customerOrders.Tables["Customers"].Rows) {

Console.WriteLine(custRow["CustomerID"].ToString());
foreach (DataRow orderRow in custRow.GetChildRows(customerOrdersRelation)) {

Console.WriteLine(orderRow["OrderID"].ToString()); }

}
```

SLIDE 20

Entity Framework is an object-relational mapper (O/RM) that enables .NET developers to work with a database using .NET objects. It eliminates the need for most of the data-access code that developers usually need to write.
Entity Framework is an open-source ORM framework for .NET applications supported by Microsoft. It enables developers to work with data using objects of domain specific classes without focusing on the underlying database tables and columns where this data is stored. With the Entity Framework, developers can work at a higher level of abstraction when they deal with

data, and can create and maintain data-oriented applications with less code compared with traditional applications.

C# CommandBuilder

The C# CommandBuilder in ADO.NET helps developers generate update, delete., and insert commands on a single database table for a data adapter. Each data provider has a command builder class.

The OleDbCommandBuilder, SqlCommonBuilder, and OdbcCommandBuilder classes represent the CommonBuilder object in the OleDb, Sql, and ODBC data providers. These classes also work in a similar fashion. Once you know how to use OleDbCommandBuilder, you can use SqlCommandBuilder and OdbcCommandBuilder in a similar way.