The Benefits of ADO.NET

in a MVC Application

BY MIKE DROZDA & DEAN CHOI

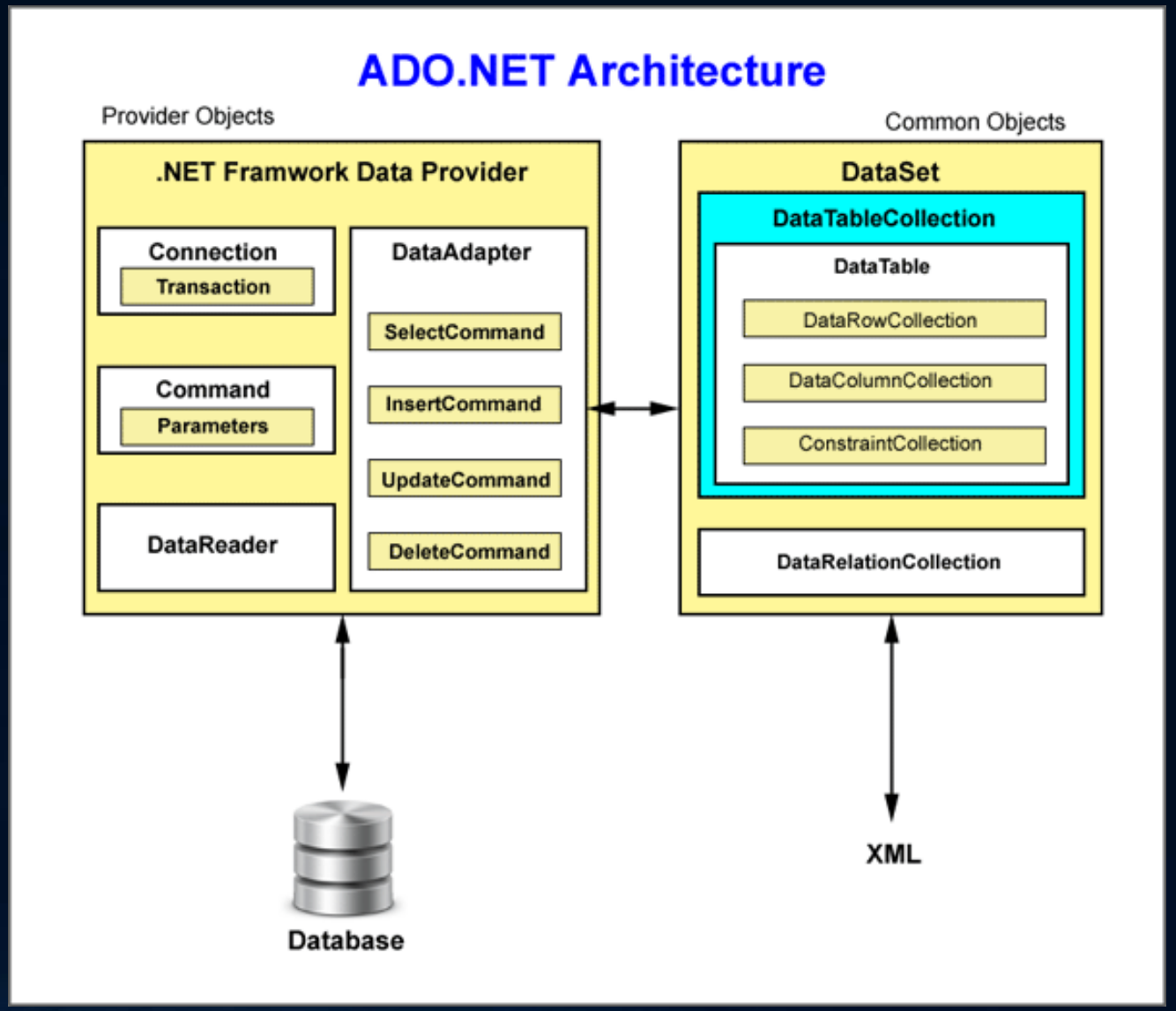
The Benefits of ADO.NET

- What is ADO.NET?
- ADO.NET Architecture
- Advantages of ADO.NET
- Steps for Reading Data From a SQL Server Database
- Demo of ADO.NET integrated into an MVC application

What is ADO.NET?

- **ADO.NET** is a set of computer software components that programmers can use to access data and data services based on disconnected **DataSets** and **XML** that is part of .NET.
- It is an evolution of **ActiveX Data Objects (ADO)** technology which was the common set of classes for interacting with databases.
- It is part of the base class library System.Data namespace.
- ADO.NET was built to serve both persistent and disconnected data needs.

ADO.NET Architecture



What are the benefits of ADO.NET?

- Interoperability
  - ADO.NET applications can take advantage of the flexibility and broad acceptance of **XML**.
- Maintainability
  - Software architects can choose to **divide** the server's business-logic processing and user-interface processing onto separate tiers on separate machines. In effect, the application server tier is replaced with two tiers, alleviating the shortage of system resources.
- Programmability
  - ADO.NET data components in Visual Studio **encapsulate** data access functionality in various ways that help you program more quickly and with fewer mistakes. ADO.NET's **robust data classes** are generated by the designer tools result in typed datasets. This in turn allows you to access data through typed programming.

What are the benefits of ADO.NET (cont'd)

- Performance
  - For disconnected applications, ADO.NET datasets offer performance advantages over ADO disconnected recordsets. When using Component Object Model (COM) marshalling to transmit a disconnected recordset among tiers, a significant processing cost can result from converting the values in the recordset to data types recognized by COM. In ADO.NET, such **data-type conversion is not necessary**.
- Scalability
  - ADO.NET accommodates scalability by encouraging programmers to conserve limited resources. Because any ADO.NET application employs **disconnected access** to data, it does not retain database locks or active database connections for long durations.

Steps For Reading Data From a SQL Server Database

- When using basic ADO.NET, there are a few steps to connecting to a SQL Server Database:
  1. Create a connection
  2. Create a command
  3. Loop through the data reader
  4. Close the reader
  5. Close the connection

Our MVC application Demo

- Goal: Display the total Order Information for a specific Order ID from the Northwind database including Company Name, Order Date, Product Details (ID, Name, Unit Price), Product Quantity, SubTotal (including discount) and Total Cost.
- Secondary Goal: Create a Display All Orders List with Total Orders Count
- Must:
  - Query the Northwind SQL Database to retrieve the data.
  - Convert and map DataReader data into C# objects and properties.
  - Display the new data in an MVC web application.

Connection String

- Connection strings tell the connection class which server to connect to, what database to use, and specified credentials (login/pwd) if necessary.

```
1   <?xml version="1.0" encoding="utf-8" ?>
2   <configuration>
3       <startup>
4           <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
5       </startup>
6       <connectionStrings>
7         <add name="Northwind"
8              connectionString="Server=MS-STDN-010\SQL2014;User=sa;Password=sqlserver;
9                                Database=Northwind;"/>
10      </connectionStrings>
11  </configuration>
```

Create Our Stored Procedure

```
Create Procedure OrderInfoByID (
    @OrderID int
) AS
SELECT o.OrderID, o.CustomerID, c.CompanyName, o.OrderDate, od.ProductID,
p.ProductName, od.UnitPrice, od.Quantity, od.Discount,
((od.UnitPrice) * (od.Quantity) * (1 - od.Discount)) AS ProdTotalCost
FROM Orders o
    LEFT JOIN [Order Details] od ON o.OrderID = od.OrderID
    LEFT JOIN Customers c ON o.CustomerID = c.CustomerID
    LEFT JOIN Products p ON od.ProductID = p.ProductID
WHERE o.OrderID = @OrderID
Go
```

Results | Messages

	OrderID	CustomerID	CompanyName	OrderDate	ProductID	ProductName	UnitPrice	Quantity	Discount	ProdTotalCost
1	10248	VINET	Vins et alcools Chevalier	1996-07-04 00:00:00.000	11	Queso Cabrales	14.00	12	0	168
2	10248	VINET	Vins et alcools Chevalier	1996-07-04 00:00:00.000	42	Singaporean Hokkien Fried Mee	9.80	10	0	98
3	10248	VINET	Vins et alcools Chevalier	1996-07-04 00:00:00.000	72	Mozzarella di Giovanni	34.80	5	0	174

FROM NORTHWIND DATABASE

Conclusion

- Any SQL Statement we can run in SQL Studio, we can also run from C# to select data.
- Commands allow us the flexibility to call statements, call procedures, and add data to our calls to pass to the database.