Disconnected Data Access Architecture using ADO.NET Framework

Vaikunth Pai T,[1] & P. S. Aithal [2]

[1,2] Srinivas Institute of Management Studies, Srinivas University, Mangalore – 575 001, INDIA

E-mail: vpaistar@yahoo.com

Type of the Paper: Research Analysis Paper.
Type of Review: Peer Reviewed.
Indexed In: OpenAIRE.
DOI: http://dx.doi.org/10.5281/zenodo.887357.
Google Scholar Citation: IJAEML

How to Cite this Paper:

Vaikunth Pai, T. & Aithal, P. S. (2017). Disconnected Data Access Architecture using ADO.NET Framework. *International Journal of Applied Engineering and Management Letters (IJAEML)*, 1(2), 10-16.
DOI: http://dx.doi.org/10.5281/zenodo.887357.

Disconnected Data Access Architecture using ADO.NET Framework

Vaikunth Pai, T. & P. S. Aithal
Srinivas Institute of Management Studies, Srinivas University, Mangalore – 575 001, INDIA
E-mail: vpaistar@yahoo.com

ABSTRACT

A significant aspect of any software application is the creation, storage, processing, transmission, and access to data. ADO.NET is the data access component for the .NET Framework of the Microsoft. This component used by .NET products to communicate with a database for recording, fetching, and updating data and also to support all type of databases. It has collections of classes, interfaces, and structures for managing the data access from different databases. ADO.NET bridges the .NET application and database communication through XML to exchange data and add many new program interfaces to access the database. This paper introduces the structure, feature and inbuilt object of the ADO.NET technology and analysis database access technique of the ADO.NET. The paper also analyses the method of the ADO.NET connection with various types of database.

Keywords: .NET Framework, ADO.NET, Data Adapter, Data Command, Data Reader, Data Provider, DataSet, ADO.NET object model, Namespaces in ADO.NET.

1. INTRODUCTION :

Access, transport, storage, and manipulation of data are fundamental and underlying processes at the heart of every information communication technology application/system. Recently introduced ADO.NET by Microsoft support these processes through different supporting run-time platforms, infrastructures, classes, levels of functionality, and part of different Microsoft strategic paradigms. ADO.NET is a part of the .NET initiative introduced in June 2000 as a new strategy for the development, deployment, and execution of highly distributed applications manage data and databases referred to as Web Services. A networking infrastructure is provided that is layered on the Internet or Intranet. This infrastructure is leveraged against technologies that are both ubiquitous and nonproprietary, and built upon industry set standards and protocols. The .NET initiative provides a completely new .NET platform including a new .NET Framework development environment and a new Common Language Runtime run-time environment. ADO.NET, by default, provides many of the benefits of being part of the .NET environment such as a foundation based on the acceptance of an object-oriented approach, coupled with a new self-describing component model called the .NET assembly. Some important benefits include a rich class library that is extensible and organised by namespace, a Common Type System, managed execution of applications, language interoperability, structured exception handling, and simple deployment.

2. ADO.NET :

ADO.NET is the data access component of the .NET framework. This model enables the communication with the databases as wee as other structures like arrays and collections also. It supports a data centric application development and uses the disconnected architecture for accessing the data. Thus it conserves the system resources and reduces the overhead of network traffic. This leverages the power of XML to provide disconnected data accessibility. It supports all types of databases and can communicate with a database for retrieving, accessing and updating the data. ADO.NET is establishing connection with a data source, send queries and update statements to the data source and returns the result.

The ADO.NET model provides a philosophical change in approach where its set of objects is functionally optimised for specific tasks. ADO.NET data access is based on the Dataset and data provider that provides specific access to a data source. The Dataset is akin to a mini-relational database but stored in memory. It is composed of a core set of objects that include its namesake, as well as the Data Table, Data Column, Data Row, Data View, Data Relation and Constraint objects. The Dataset is equated with the disconnected aspect of ADO.NET. The Data Set is a standalone object existing independently of any data source. The data provider provides the means to communicate with any data source. A .NET data provider provides a set of objects that enable the connection to, data retrieval from, data update to, and disconnection from, a data source. Each data provider is for a specific data source. Each data provider provides its own set of classes such as for SQL Server.NET, OLEDB.NET or Oracle .NET. As an example SQLServer.NET uses Sql Connection, Sql Command, Sql Parameter, Sql Data Reader, Sql Data Adapter and Sql Transaction objects (located in System.Data.SqlClient namespace) and OLE DB .NET data providers use Ole Db Connection, Ole Db Command, Ole Db Parameter, Ole Db Data Reader, Ole Db Data Adapter and Ole Db Transaction.

The ADO.NET model introduces a level of abstraction where there is a separation between data and the process by which data is retrieved and updated. The centerpiece of this model is the data adapter. Each data adapter has its own specific data provider. It is the bridge between the Data Set and the data source. It is the coordination layer between the in-memory Data Set and the permanent sources of data access through the data provider.

The data adapter uses the data reader of the same data provider to populate a Dataset using the Fill method. The data adapter uses four instances of the Command object: Select Command, Insert Command, Update Command and Delete Command, to propagate, via the Update method, the changes of the same type in a Dataset to a data source e.g. an inserted row in the Dataset is used with the Insert Command. It does not automatically include this logic. The filling of a single Data Set may be performed using multiple data adapters. The population of a Dataset is not limited only to data providers but may be sourced from user input, programmatically or XML input. The data adapter does not interact directly with the connection object but indirectly through its set of four command objects. The data adapter always leaves the state of a connection as it was, that is opened or closed, after executing. This is a default process that supports the disconnected paradigm of the ADO.NET model. The Connection object provides access to a data source either to retrieve data or to provide the pipeline for applying data changes to a data source.

.NET provides data providers. These are also referred to as managed providers such as OLE DB .NET and SQL Server .NET. The OLE DB .NET managed provider provides communication to different data sources requiring COM interoperability services of the .NET Framework to access OLE DB functionality. A managed provider such as SQL Server provides a more efficient, direct and streamlined access to the API of the database in comparison to the OLE DB provider. For example, the SQL Server .NET managed provider uses Sql Data Reader, which uses its own Tabular Data Stream protocol. This protocol is managed by the CLR to access data where SQL Server exposes its data types to conform to the .NET Framework types as well as being able to expose its data in native format. That is, when retrieving results from a managed Data Reader such as Sql Data Reader, columns are retrieved in their native data type form without requiring expensive conversions.

3. ADO.NET OBJECT MODEL :

ADO.NET renders a good support for disconnected data. The object model is based upon the standards of World Wide Consortium.

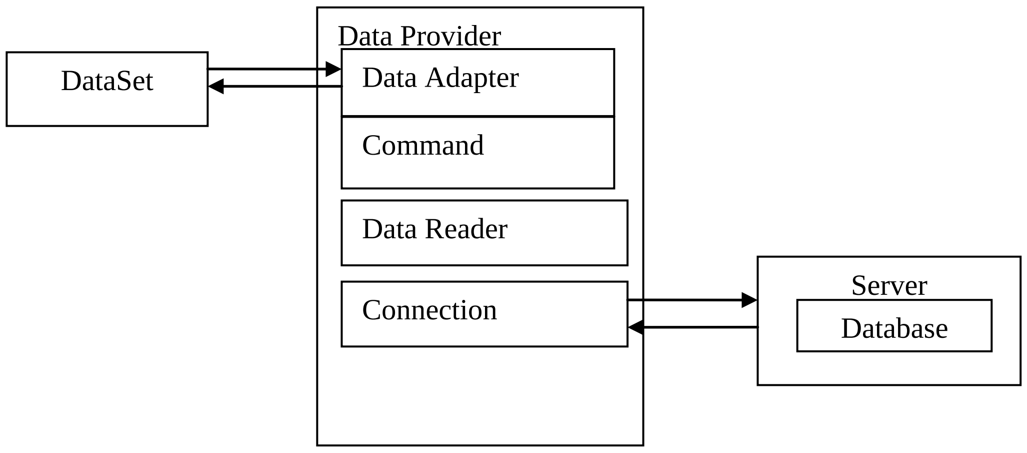

Fig. 1 : Object model of ADO.NET framework.

ADO.NET is establishing a connection with a data source, sends queries and updates statements to the data source and returns the result. Data is retrieved through a data adapter and it provides data to the application and updates the database. Data can be accessed by the application either through a dataset or through a data reader.

The key components of the ADO.NET object model [17] are discussed below:

(1) Data Store

This refers to the physical database server in the framework. They can be MS ACCESS, SQL SERVER, ORACLE or an XML file. ADO.NET providers are used to connect to these database servers.

(2) Data View

This is the representation of tables in the dataset. These are typically used for filtering or sorting the data according to user views.

(3) Data Provider

A data provider is used for connecting to a database, retrieving data, storing the data in a dataset, reading the retrieved data, and updating the database.

There are two types of data providers

a) OLEDB data provider - This type of data provider works with all the OLE DB providers, such as SQL OLE DB provider, Oracle OLE DB provider, and Jet OLE DB provider. The OLE DB Data provider classes are present in the System.Data.OleDb namespace.

b) SQL Server data provider - This type of data provider is used to work specifically with Microsoft SQL Server. A SQL Server data provider is recommended for working with a Microsoft SQL Server data source, since a SQL Server data provider allows fast access to a data source without going through an OLE DB or ODBC layer. The SQL Server data provider classes are present in the System.Data.SqlClient namespace.

Different components of the data provider:

The providers consist of Connection, Command, DataReader and DataAdapter objects for the connectivity with the database.

Connection

This component is used to establish a connection with a data source. It establishes a session with the database and manages the connection. The most common connection objects are OleDbConnection, OracleConnection and SqlConnection. OleDbConnection connects the database such as MSACCESS, DB2, OracleConnection is designed for ORACLE database. SqlConnection is specifically designed for SQL SERVER.

Data Adapter

A data adapter is integral to the working of ADO.NET since data is transferred to and from a database through a data adapter. A data adapter retrieves data from a database into a dataset and updates the

database. When you make changes to the dataset, the changes in the database are actually done by the data adapter. The data adapter first compares the data in the dataset with that in the database and then updates the database. You connect to a database by configuring a data adapter.
A data adapter uses the connection objects OleDbConnection and SqlConnection provided by the data provider to communicate with the database. Data transfer between the databases is accomplished through properties and methods of the data adapter and displaying the data is provided by table mapping.

Data Command

This object enables access to the database commands such as SQL statement or stored procedure. These commands are used to retrieve, insert, delete or modify the data. OleDbCommand object can be used to work with any OLEDB provider, SQL Command is used to access data from SQL Server and OrcaleCommand is designed for Oracle database.
For retrieving data through data commands, a Connection object is created first to connect to the database from where the data is to be retrieved. Then, a Command object is created that refers to a SQL statement or a stored procedure that is executed. The Command object can be derived from SqlCommand or OLEDbCommand class.
To access a data source, a data command should provide information about the connection for communicating with the data source, the SQL statement or the name of the stored procedure to execute, and the parameters that may be required for execution of the data command.

Data reader

Data reader is used to retrieve data from a data source in a read-only and forward-only mode. A data reader uses the Connection object to connect to the database, the Command object to execute SQL statements or procedures on the database and retrieves the data in a sequential mode. Using data reader results in faster access to data and less memory usage since at any time, only a single row is stored in the memory.

(4) Dataset

In most cases, data is retrieved through datasets, since working with data commands is very tedious and a data reader only allows data to be retrieved in a read-only and forward-only mode.
Dataset is a disconnected, cached set of records that are retrieved from a database. When a connection is established with the database, the data adapter creates a dataset and stores data in it. After the data is retrieved and stored in a dataset, the connection with the database is closed. Such a working architecture is called disconnected architecture. The dataset acts like a virtual database containing tables, rows, and columns. An application works with the database records stored in the dataset.

Types of DataSets

DataSets are classified into two types: Typed and Untyped.
• **Typed DataSet:** It is derived from the Dataset class and has an associated XML Schema. The schema contains information about the tables, columns and rows. Structure will be known at the compile time itself. The structure of typed dataset is saved in Schema Definition File (XSD). Tables and Columns are accessible by their names while programming.
• **Untyped DataSet:** It does not associate with any XML Schema and at runtime, data can be loaded into the set without specifying the structure. Tables and columns are represented as collections in the untyped set. While compiling, the structure of an untyped dataset will be unknown to the user.

4. NAMESPACES IN ADO.NET :

Namespaces used in ADO.NET are:
System.Data: It consists of classes for manipulating our data and we can represent the data logically. It also enables the user to view and share the data by using XML.
System.Data.OLEDB: It is a collection of classes for accessing the data sources such as Oracle, Ms-Access and SQL Server. It is used to establish a connection between datasets and databases.
System.Data.Common: It consists of classes that can be shared by all the .NET framework providers. These classes are providing the flexibility for the developers to write a code that will work with all .NET framework data providers.

System.Data.SqlClient: It provides managed provider for SQL Server. It is built on Tabular Data Stream to increase the performance. It consists of classes for error handling and connection pooling.
System.Data.SqlTypes: It provides classes for native SQL Server data types.

5. CONCLUSION :

ADO.NET provides a universal data access for all types of database. It uses disconnected data architecture and XML format for data transfer. The .NET Framework includes the DataSet object to work directly with the data. The object gives the flexibility and control to represent the data in users required format. In this component, the object can easily be controlled in either Windows Forms or Web Forms together with the dataset. This supports to build data entry screens quickly. The .NET Framework also includes two data providers that are used to access data sources: the OLE DB .NET Data Provider and the SQL Server .NET Data Provider. Using the components like Connection, Command, DataReader, and DataAdapter objects included in each data provider, it is possible to access the data completely as per requirement. Thus the integration of .NET with XML allows the data in ADO.NET can easily be portable and persisted locally.

REFERENCES :

[1] Adya, A., Blakeley, J. A., Melnik, S., & Muralidhar, S. (2007, June). Anatomy of the ado. net entity framework. In Proceedings of the 2007 ACM SIGMOD international conference on Management of data (pp. 877-888). ACM.

[2] Wigley, A., Sutton, M., Wheelwright, S., Burbidge, R., & Mcloud, R. (2002). Microsoft. net compact framework: Core reference. Microsoft Press.

[3] Blakeley, J. A., Campbell, D., Muralidhar, S., & Nori, A. (2006). The ado. net entity framework: Making the conceptual level real. ACM SIGMOD Record, 35(4), 32-39.

[4] Ren-peng, S. U. N. (2010). Research application of ADO .NET in multi-mode. Computer Engineering and Design, 16, 023.

[5] McManus, J. P., & Kinsman, C. (2001). C# Developer's Guide to ASP. NET, XML and ADO. NET. Addison-Wesley Longman Publishing Co., Inc..

[6] El Haddad, I. A., & Naser, S. S. A. (2017). ADO-Tutor: Intelligent Tutoring System for leaning ADO. NET.

[7] Wildermuth, S. (2003). Pragmatic ADO. NET: Data Access for the Internet World. Addison-Wesley Professional.

[8] Mackman, A., Brooks, C., Busby, S., Jezierski, E., Hogg, J., Leibovitz, R., & Campbell, D. (2003). .NET data access architecture guide'.

[9] McClure, W. B., & Beamer, G. A. (2005). Professional Ado. Net 2 Prog. With Sql Server 2005. John Wiley & Sons.

[10] Talbot, D., & Chand, M. (2008). Applied ADO. NET: Building Data-driven Solutions. Apress, (www.apress.com)

[11] Aithal, P. S., & Pai, T. (2016). Concept of Ideal Software and its Realization Scenarios.

[12] Jia, W. S. W. K. X. (2005). Use SQL Server .NET Provider Connection Pooling Effectively in ASP .NET [J]. Computer & Digital Engineering, 11, 022.

[13] Lendvai, A. J., & Shi, H. (2007). ADO and ADO .NET Object Model Comparisons: A Relational Perspective. IJCSNS International Journal of Computer Science and Network Security,, 7(1), 330-337.

[14] Chaudhuri, S., Narasayya, V., & Syamala, M. (2007). Bridging the application and DBMS profiling divide for database application developers. In Proceedings of the 33rd international conference on Very large data bases (pp. 1252-1262). VLDB Endowment.

[15] Crous, T., Danzfuss, T., Liebenberg, A., Moolman, A., & Hub, I. (2005). Adaptive Object Modelling using the .NET Framework. .NET Technologies 2005, 177.

[16] Vaikunth Pai, (2016). ADO and ADO.NET Database Access Technology Comparisons. *International Journal of Current Research and Modern Education (IJCRME)*, 1(1), 262-268.

******