



Solution Objects White Paper

This white paper outlines the functionality being introduced in mv.NET's new "Solution Objects" component set.

Introduction

The mv.NET product provides a 100% native .NET interface to all MultiValue database platforms, allowing .NET developers to access all aspects of MultiValue systems – both data and program code – from within their .NET application.

The mv.NET architecture has been designed with both performance and flexibility in mind. This, combined with an implementation that provides seamless integration with the .NET environment, provides a powerful tool for enabling MultiValue developers to harness the full power of both their MultiValue system and the .NET platform.

The new Solution Objects component set adds a number of new important advanced features to the existing mv.NET product:

- Strongly-typed, class (business object) based access to the underlying MultiValued database via a new entity model definition and code generation tool hosted within the existing mv.NET Data Manager utility
- Support for bi-directional native .NET data binding via support for the standard Visual Studio datasource interface
- Seamless integration of non-Multivalued data sources (e.g. SQL) into one consolidated entity-based access layer
- Multiple user views (business access layers) to the same underlying data

Solution Objects Concept Overview

Solution Objects provides both design-time and run-time functionality.

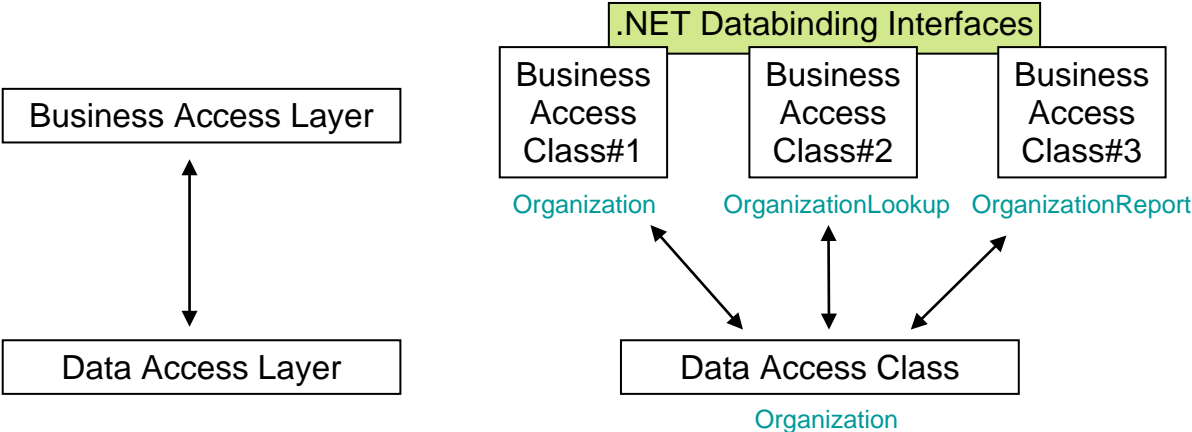
Design-Time Features

The Data Manager utility is enhanced with new entity modeling and code generation capabilities.

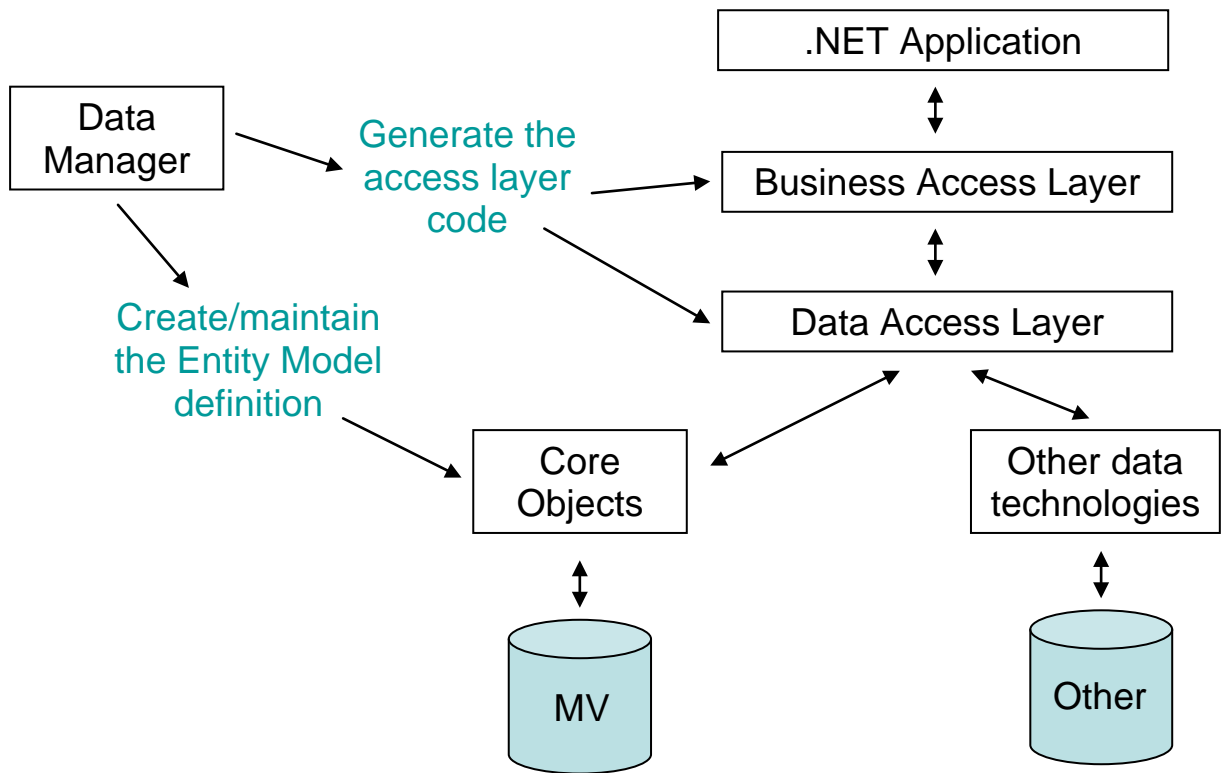
The developer is able to define the existence and content of application "entities", for example, an Organization entity. These entity descriptions are initially based on the mv.NET Extended Dictionary definitions, but are capable of encapsulating many advanced pieces of data shape and relationship information.

The entity modeling is fully nested-data aware in that it understands the concept of multi/sub-valued data and the data maintenance implications of such data structures.

Two levels of entity modeling are supported: the Data Access Layer (DAL) and the Business Access Layer (BAL). The DAL holds a complete definition of the entity-to-database mapping process. The BAL exposes a variety of sub-sets of the DAL. The purpose of the BAL is to provide customized "views" of the same underlying entities in order to impose a simplified or tighter level of security layer for different groups of developers. The BAL is also the place where .NET data binding is implemented.



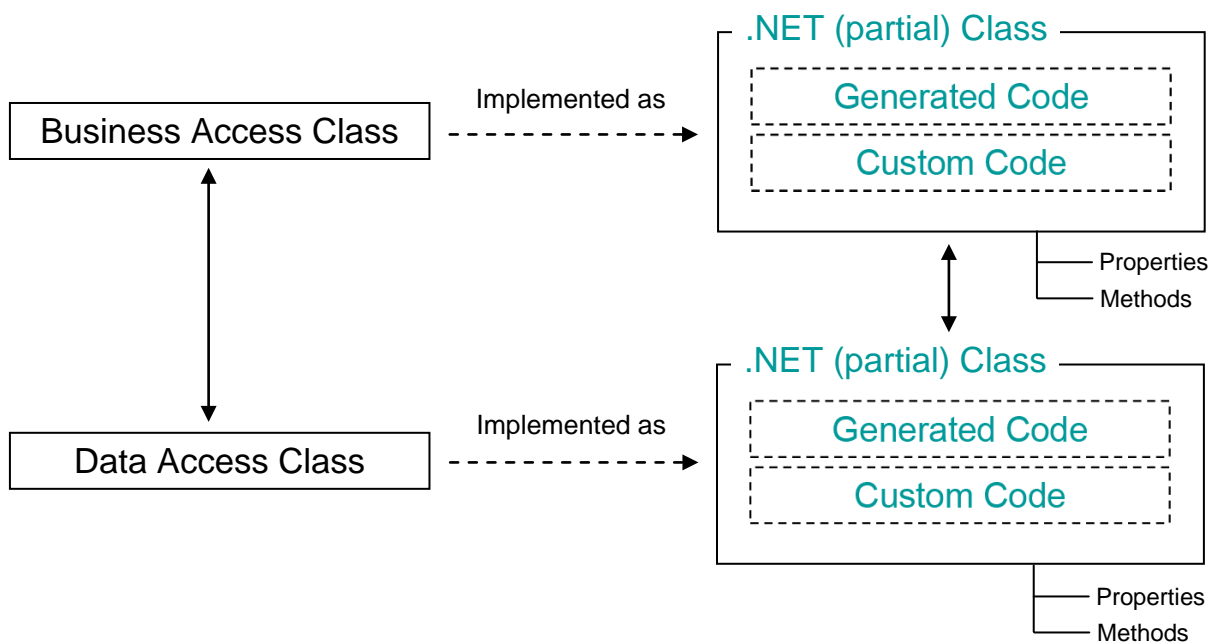
Once the entities within an application have been defined, the Data Manager allows the developer to generate a set of code modules which implement the entity definitions.



Run-Time Features

Once the Data Manager has generated the DAL and BAL code, the developer is able to incorporate this code into one or more Visual Studio projects and create assemblies for use by the developer community.

The DAL and BAL expose the defined entities as a series of standard .NET partial classes:



The use of partial classes allows the developer to customize and enhance the classes as necessary with whatever extra functionality is relevant/required for the application.

The properties on each class typically expose database-resident information. The Methods expose selection/update/delete functionality along with links to back-end databasic subroutines.

The generated code makes use of many of the advanced features of mv.NET's current Core Objects component set.

Access to other data sources is achieved by the use of other data access technologies, for example ADO.NET providers for MS SQL.

Visual Studio Datasource Support

The Business Access Layer can be used as a standard Visual Studio object datasource, allowing any 3rd party tool or component full read/write access to the underlying MultiValue database.

The code generation feature of the Data Manager automatically adds all of the interface implementation and class decoration code into the generated classes for the support of standard .NET databinding. Thus, the developer is able to immediately use any of the Business Access Classes (BACs) with .NET databinding for both Web and Rich-client applications.

Feature Summary

Solution Objects provides the following key features

- The generation of a strongly-typed access layer into the MultiValued database
- Abstracts all database interaction into a Data Access Layer (DAL) and a Business Access Layer (BAL)
- Code generation based on current extended dictionary definitions and further "Entity Model" details
- Full support for bi-directional Win and Web databinding
- Support for multiple data sources, including SQL
- Fully nested-data aware
- Integral and program-controllable fetch-on-demand ("lazy") data loading
- Stateless or Stateful connections with automatic database-hosted data persistence for easy web application/service application development
- Optimistic and Pessimistic locking support
- Multivalue and Subvalue positional integrity during all data maintenance operations
- Strong integration with back-end databasic subroutine code
- Fully extendible generated code via the standard .NET partial class mechanism

Product Applicability

Solution Objects is relevant for the MultiValued developer community for the following reasons:

- Provides intuitive, easy to use database access for developers who don't know MultiValued database technology or databases in general
- Concentrates (encapsulates) MultiValued database knowledge/know-how in the DAL
- Strongly-typed error detection at compile-time with full Visual Studio intellisense and XML help support
- Encourages developers to produce more intuitive, readable code
- Provides full bi-directional support for native .NET databinding down to the sub-value level
- Full integration with thousands of 3rd party products and components via the Visual Studio datasource mechanism
- Faster, more effective application development
- Allows multiple, heterogeneous datasources within an organization to be accessed through a single unified entity model