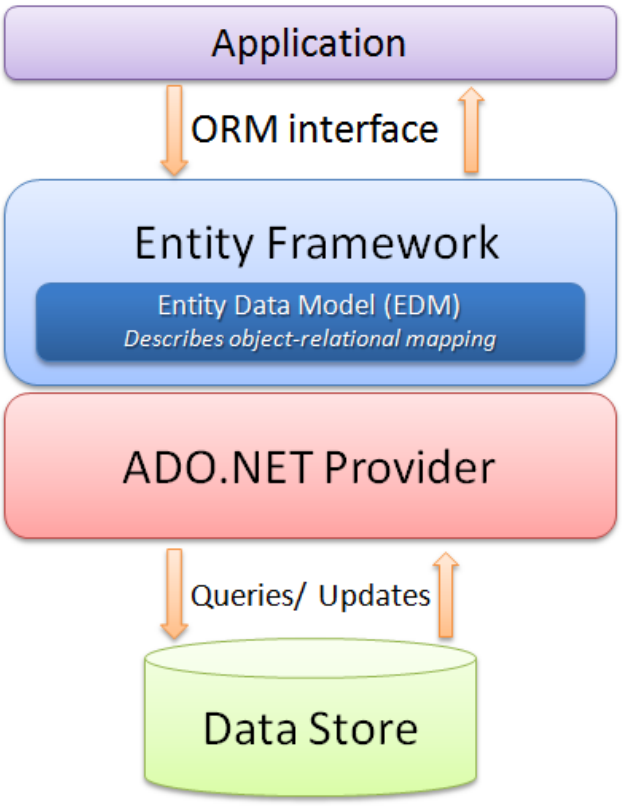
ADO.NET Entity Framework At-a-Glance

The Microsoft® ADO.NET Entity Framework is an Object/Relational Mapping (ORM) framework that enables developers to work with relational data as domain-specific objects, eliminating the need for most of the data access plumbing code that developers usually need to write. Using the Entity Framework, developers issue queries using LINQ, then retrieve and manipulate data as strongly typed objects. The Entity Framework's ORM implementation provides services like change tracking, identity resolution, lazy loading, and query translation so that developers can focus on their application-specific business logic rather than the data access fundamentals.

High-level capabilities of the Entity Framework:

- Works with a variety of database servers (including Microsoft SQL Server, Oracle, and DB2)
- Includes a rich mapping engine that can handle real-world database schemas and works well with stored procedures
- Provides integrated Visual Studio tools to visually create entity models and to auto-generate models from an existing database. New databases can be deployed from a model, which can also be hand-edited for full control
- Integrates well into all the .NET application programming models including ASP.NET, Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), and WCF Data Services (formerly ADO.NET Data Services)

The Entity Framework is built on the existing ADO.NET provider model, with existing providers being updated additively to support the new Entity Framework functionality. Because of this, existing applications built on ADO.NET can be carried forward to the Entity Framework easily with a programming model that is familiar to ADO.NET developers.



Using the Entity Framework to write data-oriented applications provides the following benefits:

- Reduced development time: the framework provides the core data access capabilities so developers can concentrate on application logic.
- Developers can work in terms of a more application-centric object model, including types with inheritance, complex members, and relationships. In .NET Framework 4, the Entity Framework also supports Persistence Ignorance through Plain Old CLR Objects (POCO) entities.
- Applications are freed from hard-coded dependencies on a particular data engine or storage schema by supporting a conceptual model that is independent of the physical/storage model.
- Mappings between the object model and the storage-specific schema can change without changing the application code.
- Language-Integrated Query support (called LINQ to Entities) provides IntelliSense and compile-time syntax validation for writing queries against a conceptual model.

The Entity Framework uses the Entity Data Model (EDM) to describe the application-specific object or "conceptual" model against which the developer programs. The EDM builds on the widely known Entity Relationship model (introduced by Dr. Peter Chen) to raise the abstraction level above logical database schemas. The EDM was developed with the primary goal of becoming the common data model across a suite of developer and server technologies from Microsoft. Thus an EDM created for use with the Entity Framework can also be leveraged with WCF Data Services (formerly ADO.NET Data Services), Windows Azure Table Storage, SharePoint 2010, SQL Server Reporting Services, and SQL Server PowerPivot for Excel, with more coming in the future.