The Microsoft **ADO.NET Entity Framework** (**EF**) is the preferred Microsoft technology for **relational database access** as well as entity data modeling.

Entity Framework includes a data access layer (DAL) that translates client requests to retrieve (query) and save entities into commands that the database understands. In practice that usually means translating LINQ queries into SQL queries and "submit changes" commands into SQL insert, update, and delete commands.

The Entity Framework is a two-tier technology and its clients must run on the full .NET 4.5 framework. A Silverlight or Windows Store application cannot use the Entity Framework.

This means that Silverlight and Windows Store clients must delegate persistence requests to a middle tier intermediary which is able to talk to EF. DevForce provides that intermediary in its *EntityServer* class. DevForce also provides to these clients a full entity model and supporting client libraries that do not refer to EF assemblies.

Back on the server, DevForce manifests client entities and persistence requests as Entity Framework entities and persistence requests. The DevForce interactions with EF require an entity data model (EDM) in the form an XML file (the edmx).

EF Code First

Most DevForce application developers prefer the Database First and Model First approaches that produce an EDMX. Developers who'd rather concentrate on the model and classes and avoid struggling with an EDMX prefer a Code First style. DevForce supports them all.

NHibernate and non-relational data sources

If you are an NHibernate developers can use DevForce POCOs and write the persistence operations yourself. You might do the same for non-relational data sources such as external web services. You'll still benefit from DevForce with its simplified WCF configuration, security, and its client-side programming model with caching and entity-oriented semantics.