DevForce developers build data-driven applications from end-to-end, from the database through to the end-user's screen using an entity-oriented development paradigm.

In that paradigm, different layers of the application exchange entity class objects that encapsulate entity identity, long-lived ("persistent") data values, relationships among entities, and business logic governing data integrity, data access rules, and workflow.

This topic describes how you **model entities** in DevForce as **.NET entity classes**. We use the word "model" as a *verb* because our attention is on the ways you produce and maintain entity classes: the tools you use, the steps you follow, and the structure of the classes themselves.

The end result is a coherent collection of entity classes representing the domain of the application, the "entity class model" ... "model" as a *noun*. The entity classes in this model are specifically designed to cross a network and participate directly in remote client UIs. The developer can and usually does extend these entity classes with application-specific, business logic

DevForce supports three essential styles of entity model development:

1. Generate the entity class model and maintain it with the Visual Studio Entity Data Model (EDM) Designer.
2. Write the entity classes and map them to the database by hand, entirely in code, using Entity Framework "Code First".
3. Write the entity class model (and often the data access layer as well) using a technology other than Entity Framework.

The third style is almost completely open ended with respect to the class coding requirements, type of remote storage, data access methods, server infrastructure, and tooling. It makes few assumptions about the nature of the entity classes themselves ... and requires the most effort to achieve what the other styles deliver almost effortlessly. We call this the POCO ("Plain old CLR object") style and we cover it elsewhere.

The topic you are reading now is devoted to the first two styles in which Entity Framework is an essential enabling technology and DevForce client application infrastructure is embedded in the entity classes themselves. Such classes - whether maintained with an EDM or handwritten "Code First" - can satisfy diverse needs on both server and client: they can track their own changes, participate in data bindings, validate property inputs automatically, be validated on demand, and suggest to a UI how they should be displayed.

You can learn how to **create and maintain** an entity model with these wide-ranging capabilities by diving into the material in this topic.