CodeFluent Entities & the ADO.NET Entity Framework

Understanding relative positioning

R&D Team

September 2014

Table of Contents

1

Introduction

I. Introduction: document objectives

As a software vendor providing a complete solution for developing .NET applications including the production of a business layer connected to the database, we keep getting questions about **how CodeFluent Entities relates to the Microsoft® ADO.NET Entity Framework.**

As a matter of fact, we see more and more messages emphasizing on the ADO.NET Entity Framework being the ultimate data access solution in the Microsoft .NET space. Field reality is a bit different as there are today many ways to access data in .NET, the most used and proven one for business applications being simply direct ADO.NET, on which all tools eventually rely.

As object-relational mapping techniques have become popular, the ADO.NET Entity framework is likely to become the most popular ORM in the Microsoft platform. However, ORMs tools are only focused on data access. Beyond that, you will have to face a lot of technical challenges to deliver applications that will perform, scale and serve your business users in an efficient way.

Those tools abstract and hide the database from the developer. Entity Framework is an ORM. CodeFluent Entities is a great alternative which provides object-relational continuity and automation of both .NET code and database scripts without hiding it or losing control.

The following parts give you more details from a technical stand point both on CodeFluent Entities and the ADO.NET Entity Framework, so one can better understand the relative positioning and measure the difference of breadth and depth between the two.

II

Overview

II. Overview

1. What is the ADO.NET Entity Framework?

"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

Provides a Code First experience to create entity models using code. Code First can map to an existing database or generate a database from the model.

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."

Source: http://msdn.microsoft.com/en-us/data/aa937709

2. What is CodeFluent Entities?

CodeFluent Entities is a unique product, integrated into Visual Studio 2008/2010/2012/2013, which allows developers to generate components such as database scripts (e.g. T-SQL, PL/SQL, MySQL, Pg/SQL), code (e.g. C#, VB), web services (e.g. WCF, JSON/REST) and UIs (e.g. Windows 8, ASP.NET, MVC, SharePoint, WPF).

The code generation process is model-first and continuous: from your declarative model, a meta-model will be inferred which code generators will then translate into code. Over 20 code generators (a.k.a. "producers") are provided "out of the box" and can be combined to create your own application following your desired architecture, using your desired technologies.

Since your application development is driven from this model, your business logic is decoupled from the technology and allows you to absorb changes faster and smoother: apply changes in your model to update all your layers consistently or add/switch technology by changing your used code generators.

CodeFluent Entities is designed for the .NET platform and empowers users to streamline developments on major Microsoft platforms and technologies such as SharePoint, SQL Server, C#, VB.NET, WCF, ASP.NET, WPF, and more.
The product now supports
non-Microsoft technologies by allowing users to generate SQL code for Oracle, MySQL, and PostgreSQL databases.

CodeFluent Entities can be seen as a superset of what the Entity Framework provides (although CodeFluent Entities does not rely on any part of the Entity Framework).

Using CodeFluent Entities, architects define business models in a centralized place; choose target platforms or technologies (including, but not limited to, data access), generate, and do it again continuously, as much as needed, without losing existing data.

In practice, developers using CodeFluent Entities mostly write business-focused code or custom user interfaces, leveraging the strongly-typed objects and components that were produced automatically by the tool, as well as the numerous runtime library classes and UI controls provided out-of-the box.

High-level capabilities of CodeFluent Entities:

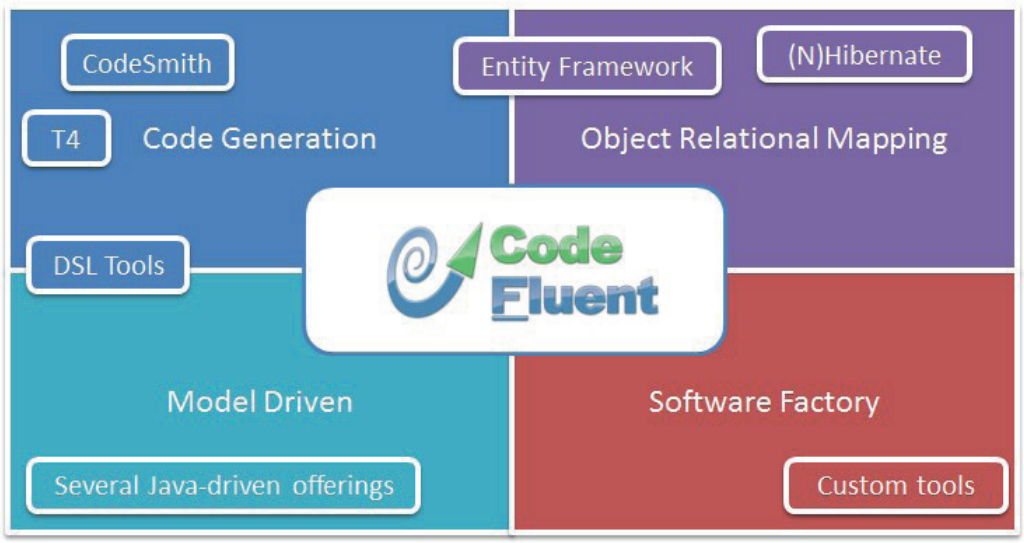
- Defines more than 40 platform-independent concepts for application design (Entity, Property, Methods, Forms, Rules, Instances …).

- Generates automatically platform-dependent files from these concepts: database schemas, tables, columns, constraints, views, stored procedures, and row data; the corresponding .NET data access layer; WCF services, configuration and enhanced proxies; Rules; Documentation; UI artifacts.

- Works and integrates with a variety of technologies (including Microsoft SQL Server, SQL Azure, Oracle Database, ASP.NET, SharePoint, Excel, WPF, Silverlight, Windows Forms, WCF)

- Provides integrated Visual Studio tools to visually create models, to import models from existing models (database, UML diagram, and Entity Framework models), and to generate all platform-dependent files from these models.

- Allows architect to define model “Aspects” to define behaviors and constraints that can be applied to concepts. As an example, the “localization aspect” is provided out-of-the-box and allows the creation data localization columns in the database without any extra work.



3. In a nutshell

The ADO.NET Entity Framework is an ORM and is the new Object Oriented data access method provided by Microsoft, on top of ADO.NET. As its full name suggests, the ADO.NET Entity Framework focuses specifically on the data access part, and as of today, especially on Microsoft SQL Server. Other databases implementations are provided only by third parties, with various level of support.

While it is a great data access tool, developers will need to overcome much more difficulties than accessing data to deliver an enterprise-class application and that is where CodeFluent Entities comes into play.

The idea is that CodeFluent Entities will generate all necessary but recurrent and low value code such as architecture related code (it can generate all layers and the plumbing between one another), but also complete application features such as application level security, internationalization, caching, search, data validation, and much more, without extra lines of code or the need to use extra tools or technologies from other vendors.

In the end, more than a data access tool, CodeFluent Entities allows you to control code production of all layers and application wide, which in turn structures teamwork, minimizes risks, and creates predictability. Furthermore, thanks to its centralized pragmatic and platform independent model, your business logic is decoupled from technology and lets you absorb functional changes smoothly.

CodeFluent Entities does not internally rely on any ORM or third party tool. It is solely based on ADO.NET. However, for specific scenarios where you would absolutely need dynamic SQL generation, CodeFluent Entities ships out-of-the-box with two specific producers:

1) A “Linq-To-Sql producer” that is capable of adding Linq-To-Sql specific custom attributes to the generated classes, automatically.

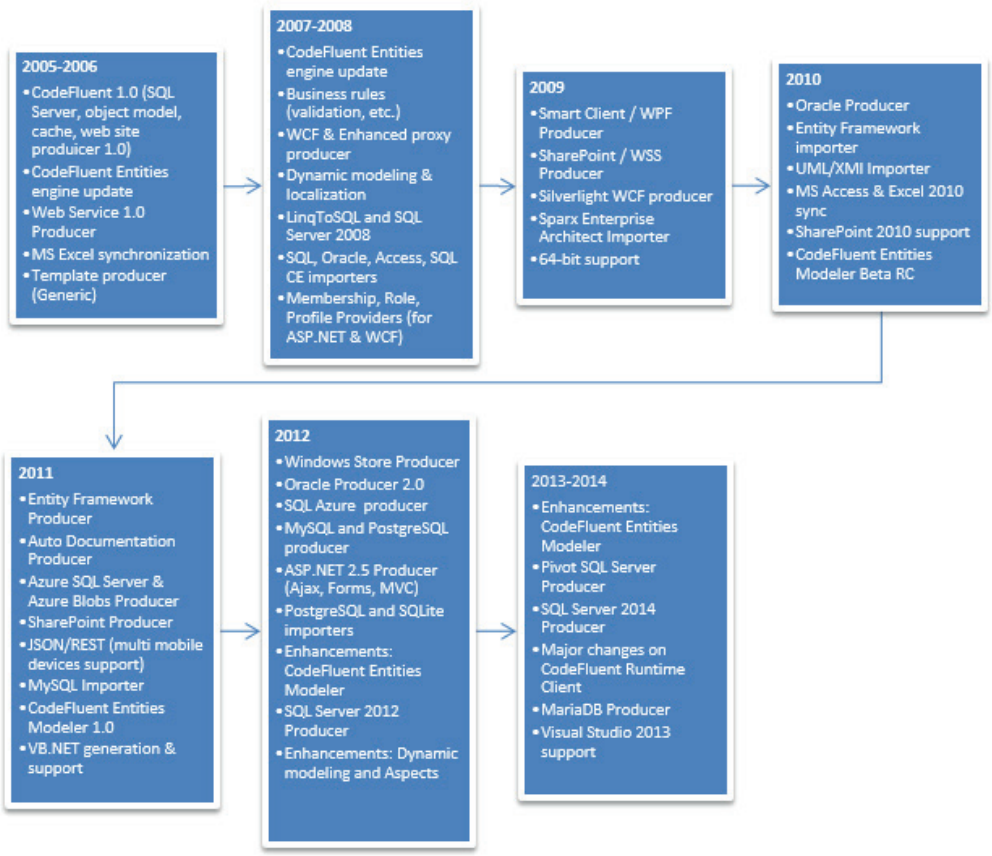
2) An “Entity Framework producer” that allows you to generate an EDMX model from your CodeFluent Entities model.

An Entity Framework importer is also available, which allows to reuse the model already define into a legacy Entity Framework model (EDMX).

4. Past and Present Roadmap

CodeFluent Entities is on the market since 2005. Customers proven product with enterprise class applications in production since years. They already made technological jumps thanks to generation approach.

The following roadmap (updated end of June 2014) shows the technological area covered by the product:



5. What Entity Framework provides

"Entity Framework allows you to create a model by writing code or using boxes and lines in the EF Designer. Both of these approaches can be used to target an existing database or create a new database. This short video explains the differences and how to find the one that is right for you."

Source: http://msdn.microsoft.com/en-us/data/aa937709

"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"

Source: http://msdn.microsoft.com/en-us/data/aa937709

6. What Entity Framework provides

Obviously the Entity Framework has a small overlap with CodeFluent Entities regarding data access features. This being said, other than data-access, CodeFluent Entities provides you with other key points such as:

- **Out-of-the-box application blocks:** security, sorting, paging, data binding, internationalization, caching, search, binary large object streaming, office integration, etc.

- **Code production control:** more control on time and costs while ensuring a homogeneous code quality, minimize heavy ground/foundation work and internal "frameworks" development.

• Structure teamwork: ensure programming consistency across the developer team; implicitly define how things must be done and avoid the “Superman syndrome”.

• Absorb functional changes: accept functional changes smoothly thanks to the continuous generation process. Improve your flexibility and strengthen your business partners’ relationships.

• Decouple from technology: secure your investments towards technology shifts, adding different target platforms to your project.

On a purely technical point of view, in addition to SQL Server data access, technologies leveraged by CodeFluent Entities are:

• Oracle Database,
• MySQL,
• SQL Azure,
• WCF,
• ASP.NET WebForms & MVC
• Silverlight,
• WPF
• SharePoint,
• Office
• MySQL
• PostegreSQL
• MariaDB
• SQLite

Moreover, architecturally speaking, without developing them yourself or using extra tools, the Entity Framework does not support layers other than the database and the object model. This being said, CodeFluent Entities can generate the following layers in addition (100% functional):

• The service layer,
• The user interface layer,
• And the most important: the built-in consistency between all layers and technologies.

7. Visual Studio Integration

a. Entity Framework

Entity Framework 6 is integrated to **Visual Studio 2012 and 2013** through a new **project type** being an **Entity Data Model**. There are also Visual Studio 2008 and 2010 versions which use a previous version of Entity Framework but this one cannot be compared feature-wise with the 2013 version.

In practice, the Entity Data Model is embedded to the C# or VB.NET project, therefore it's not completely model-first since it's already tied to a language. Furthermore, this demonstrates the Entity Framework is more a new data access model on top of ADO.NET (remember its name is actually "ADO.NET Entity Framework"), rather than a complete solution to build new applications foundations.

b. CodeFluent Entities

CodeFluent Entities, which is on the market since 2005, is composed of two main components:

- **The CodeFluent Entities Modeler:** graphical interface fully integrated with **Visual Studio 2008/2010/2012/2013 and providing the same features from .NET 2 to 4.5.** In practice, it adds a new Visual Studio project type where you can design your model, import models and generate files using standard projects as targets from Visual Studio solution.

- **The CodeFluent Entities Meta Compiler:** a code generation engine with out-of-the-box producers generating ready-to-use code for following technologies: scripts **(T/SQL, PL/SQL, MySQL, PostgreSQL, MariaDB, SQLite), Classes C#, interfaces WCF (Silverlight, .NET), SharePoint Web Parts, WPF, Office List Synchronization, ASP.NET Web Controls & Web Site.** The human-readable generated sources implement best practices and recognized patterns, and are designed for testability, ease of maintenance and performance.

Schema References Comparison

III. Schema References Comparison

Once again, the Entity Framework is an ORM whereas CodeFluent Entities is a full model-driven software factory. Therefore, CodeFluent Entities covers much more areas than the current version of the Entity Framework does.

Here are a set of paragraphs and screenshots comparing just the number of available attributes for the main concepts:

1. Concepts Overview

a. Entity Framework

- Entity
- Property
- Complex type
- Association (1-N, N-1, N-N, inheritance)
- Function
- Enumeration
- Parameter

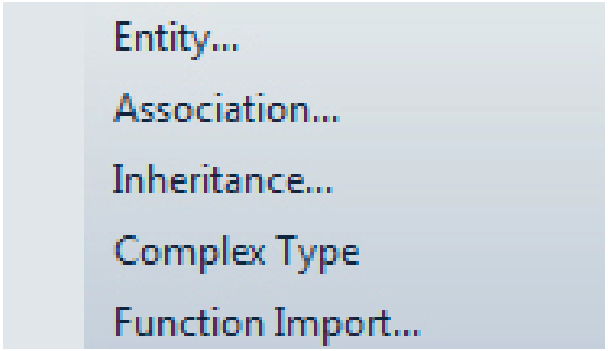
b. CodeFluent Entities

- Entity,
- Lightweight Entity (represent light objects as structs or non-persisted classes),
- Enumeration (including multi-valued),
- Property,
- Method,
- Parameter,
- Instance (represents a line of data and can be used to initialize the generated database)
- Schema (translated into actual SQL schemas by persistence producers),
- Store (models can have several data stores),
- Namespace (models can contain multiple namespaces),
- Category (used by UI producers to group entities),
- Relationship (1-1, 1-N, N-1, N-N, non-persistent, inheritance)
- UserType (define custom user data type),
- Renderer (indicates how to render something in the UI layer),
- Attribute (translated into .NET attributes by the .NET business object model producer),
- Attribute Arguments,
- Aspect (allow developers to work on the inferred model to set-up application wide changes across all layers such as enabling data localization on a project for instance),
- Rule (event, transaction, validation, authorization, implementation, custom),
- View (and persistent views),
- Message (localization, documentation, resources),
- Form (design platform independent forms which will be translated into actual forms in UI layers),
- Field (of a Column) and column (of a tab)

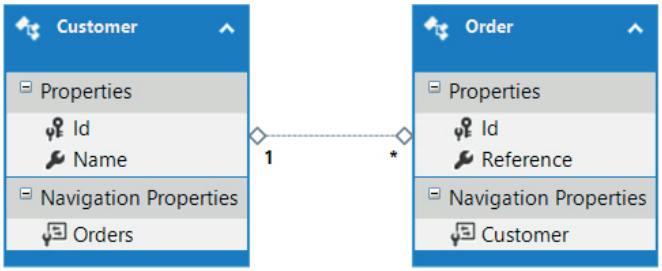
2. Key Concepts Overview

a. Entity Framework

Entity Framework only deals with data-related elements, which is normal considering its role as an ORM.

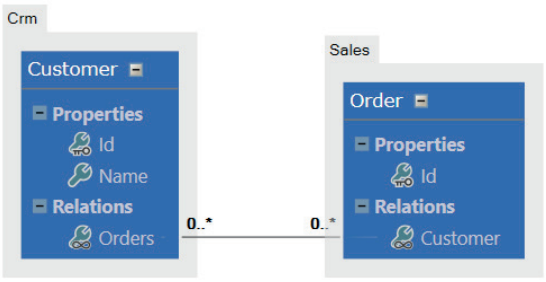


Here is the graphical layout of two related entities where we can see only properties (standard, complex or navigation):

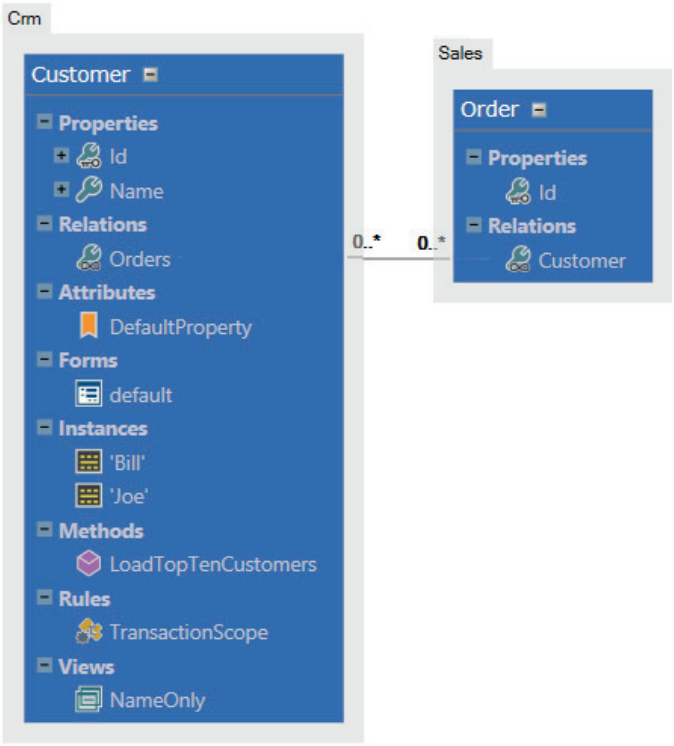


b. CodeFluent Entities

On the other hand, here is the equivalent diagram (in Modern UI style – yes the user can customize how the graphical elements are rendered on the design surface) in a CodeFluent Entities surface:



And the same when all the visually rendered concepts (some concepts are not displayed on graphical surfaces) attached to the entity are expanded:



Key points:

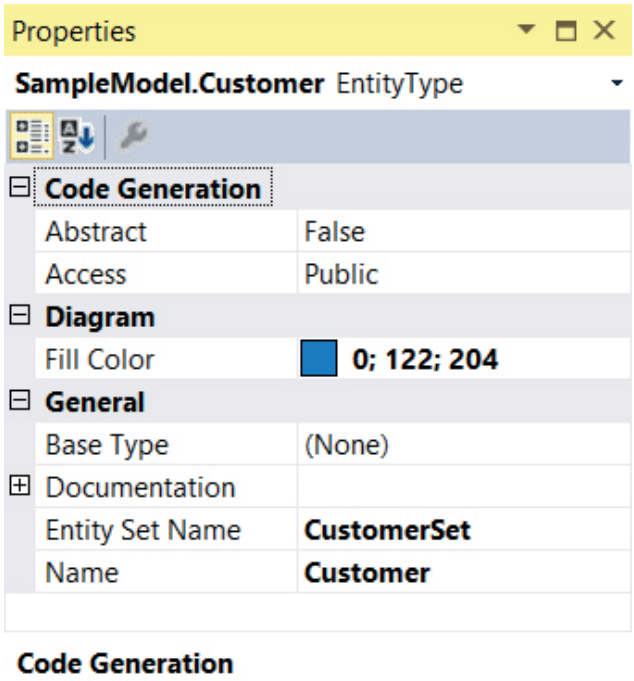
- Entities can be grouped in namespaces (two namespaces are shown here)
- The “Customer” box is in fact not a box; it’s a full tree view with all the concepts hierarchy carefully organized. For example the **Property** concept has also attributes, or rules, just like the **Entity** concept.
- The **Instance** concept is one of the exclusive concepts introduced by the product: a powerful way to describe instances of the given entity. Here, “Bill” and “Joe” will be saved as rows in the Customer table.
- The **Form** concept is a platform-independent concept that allows the user to design UI Forms that will become target-specific forms after generation (Web, WPF, etc.)

3. Entity and Property Attributes

The set of hard copies in the following pages are aimed at demonstrating the **difference in depth** between the two approaches, a difference which can easily be explained by the fact that Entity Framework is clearly positioned and limited to dealing with the «mapping» aspect of the data layer.
Please note we are only comparing the **Entity** and **Property** concepts because they are the only concepts found in Entity Framework, but CodeFluent Entities defines more than 40 concepts.

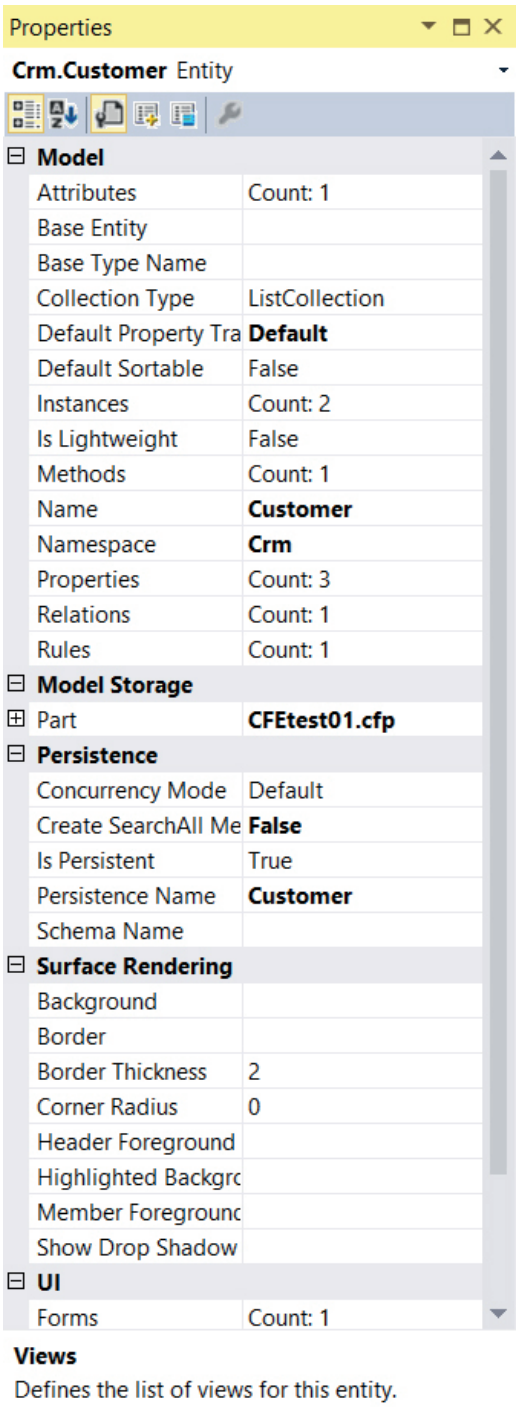
a. Entity Framework Entity

Here are the elements that you get at entity level in Entity Framework:

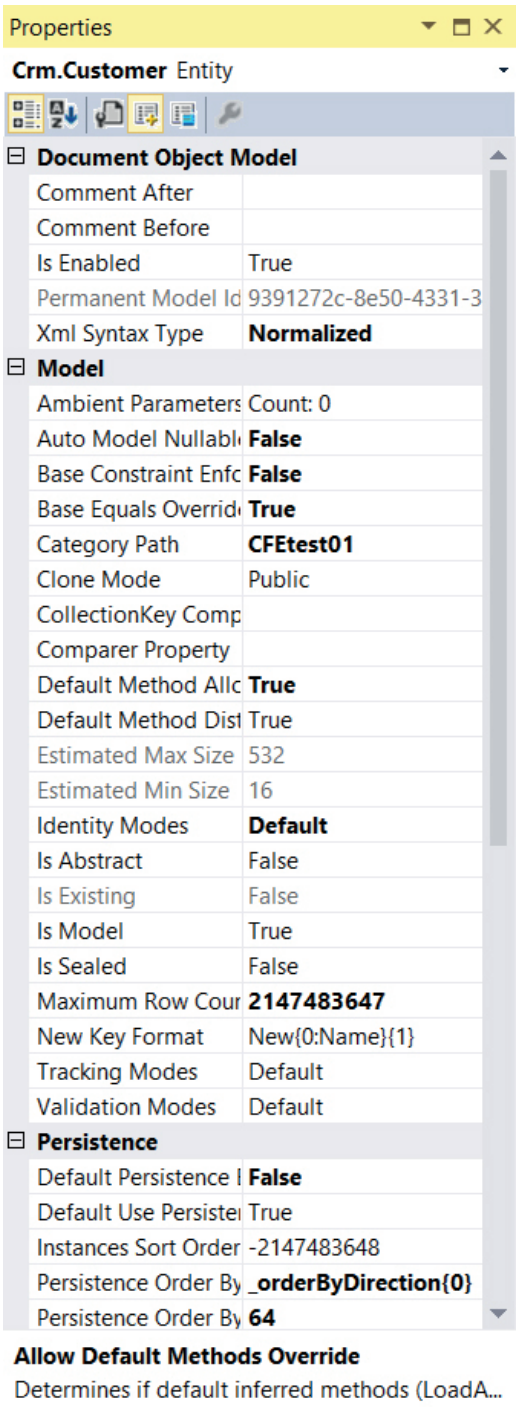


b. CodeFluent Entities

Here are the elements that you get at entity level in CodeFluent Entities (on the left the basic ones, on the right, the part of the advanced ones since it would require an extra screenshot to view them all):



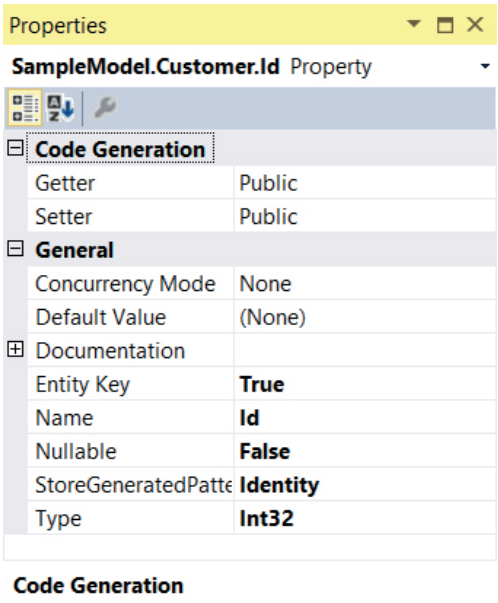
Simple View



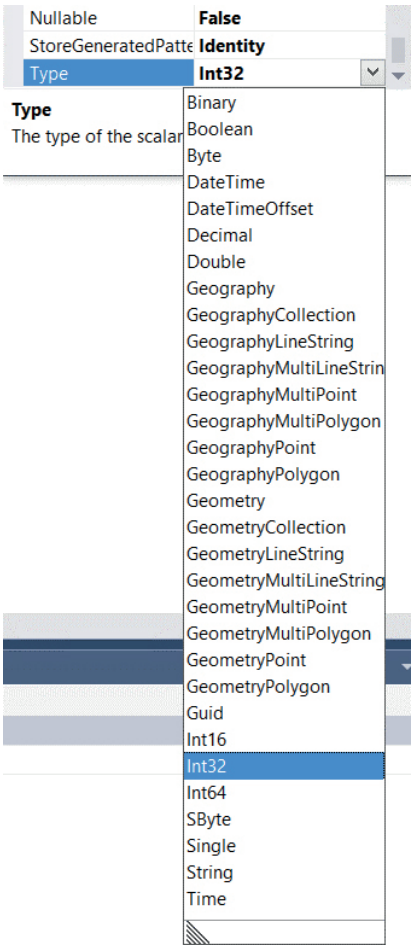
Advanced View

c. Entity Framework Property

Here are the attributes that you get at property level in Entity Framework. It's also interesting to note the Type field will contain only SQL Server types:

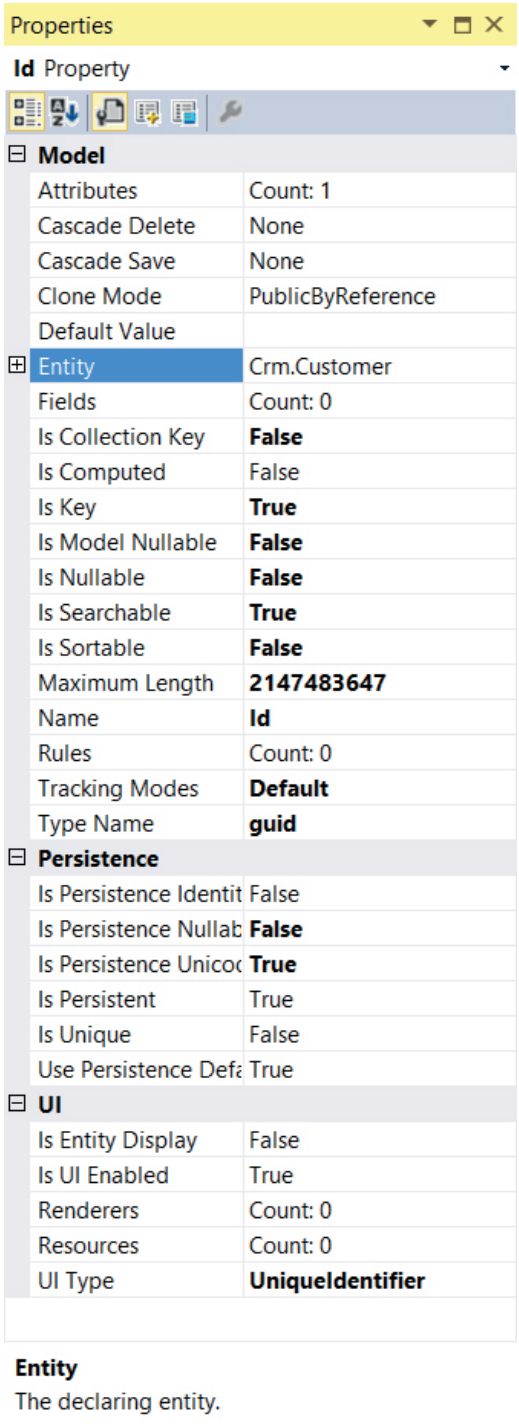


And here is the list of supported types (it matches in fact SQL Server data types, most of them are new spatial data types):

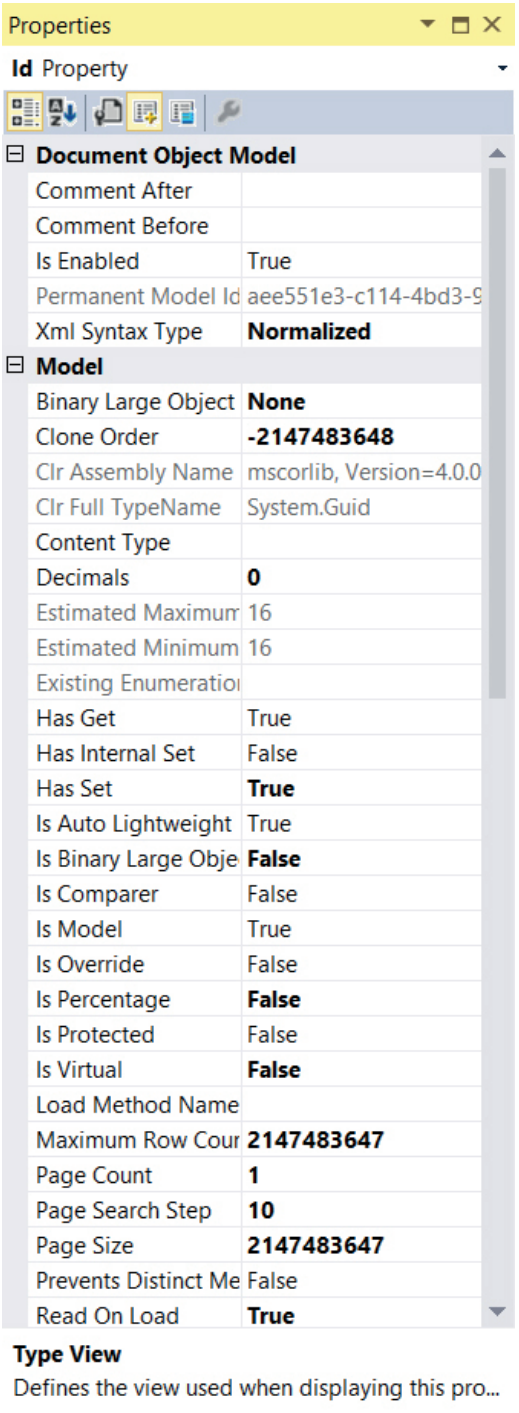


d. CodeFluent Entities Property

Here are the attributes that you get at property level in CodeFluent Entities (limited by page size, the grid property would require two pages to fit in):

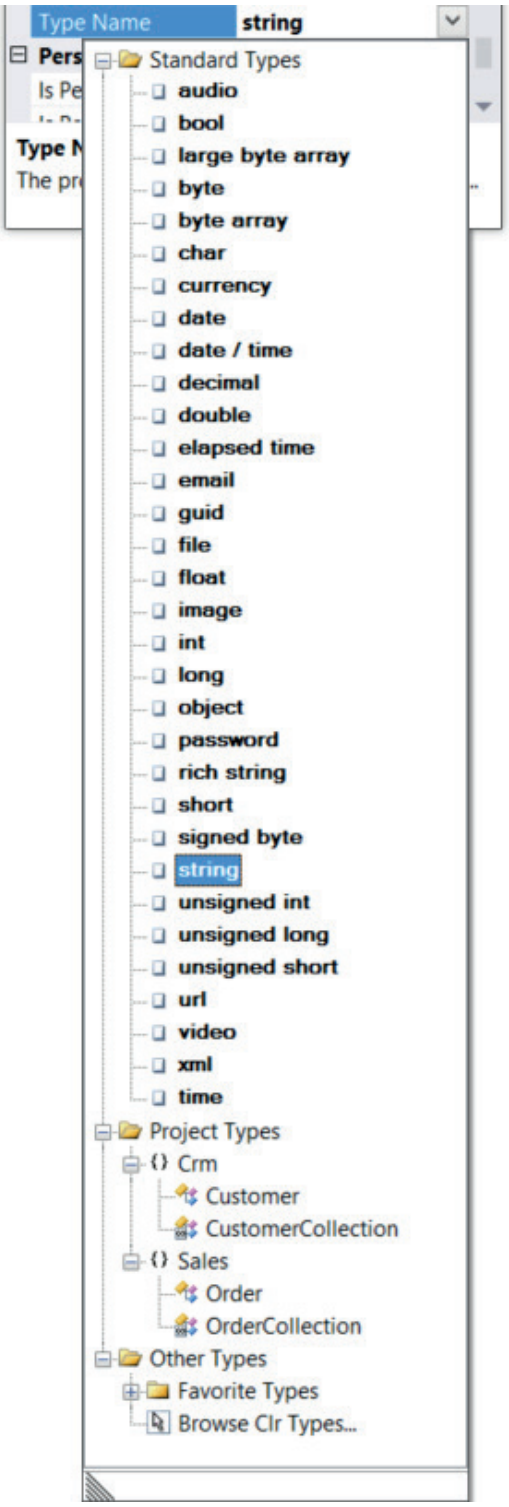


Simple View



Advanced View

And here is the list of supported types for CodeFluent Entities properties:



Please note the product also supports any serializable CLR or existing type, enumerations and Sql Server types (SqlGeography, SqlGeometry and SqlHierarchyId):
http://www.softfluent.com/forums/codefluent-entities/sql-server-2008-geography-type).

IV

Exclusive features

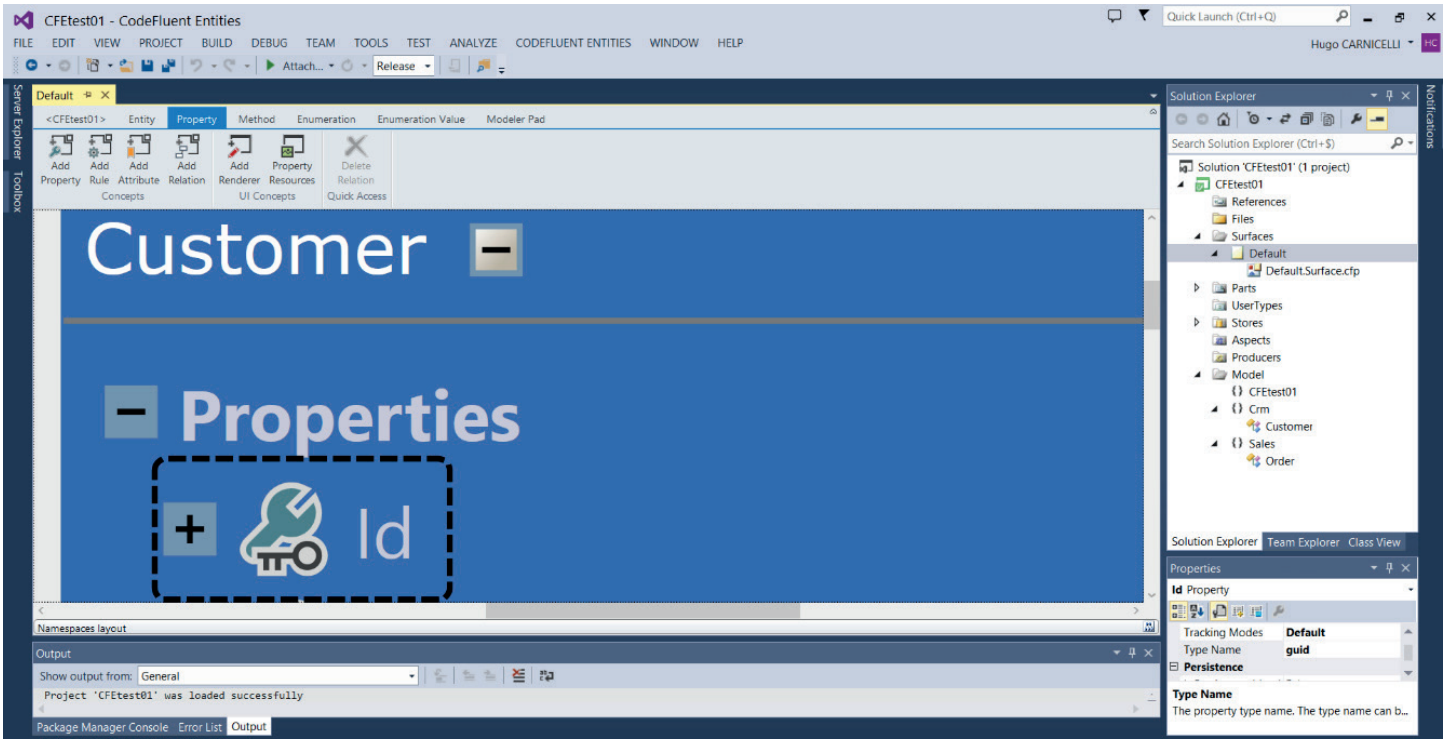
IV. Exclusive features

Here are some exclusive features of the product:

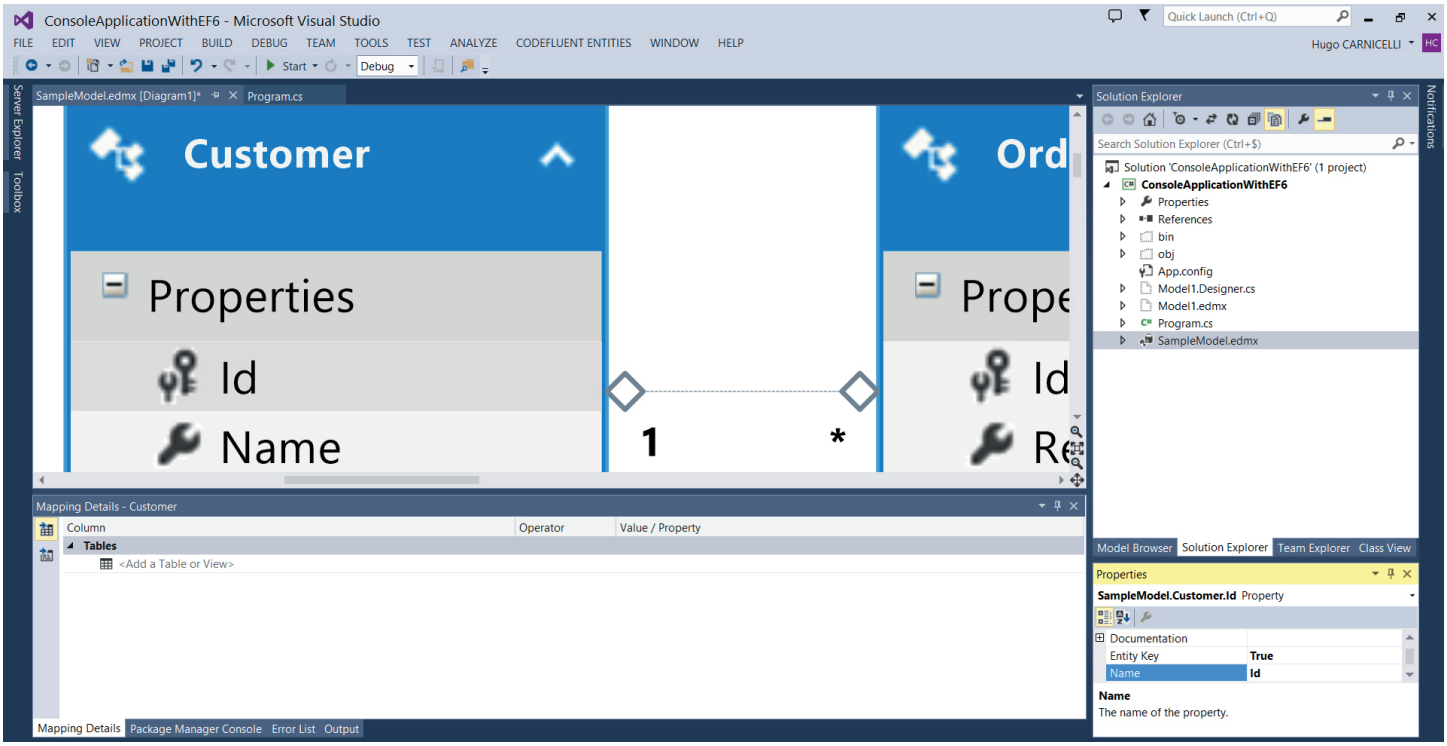
1. Graphical Editor

a. Vector Graphics

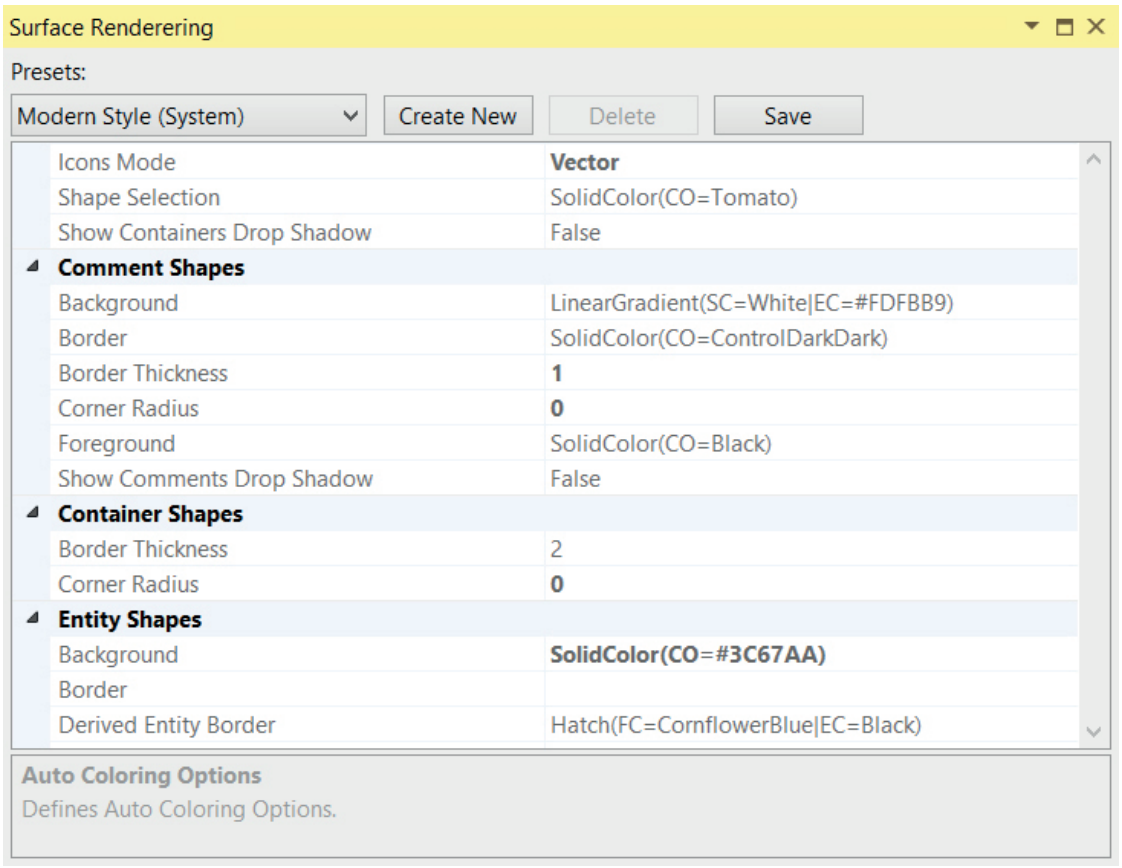
Unlike Entity Framework designer which uses the now obsolete DSL-tools technology, the CodeFluent Entities Modeler renders the living design surface using fast vector graphics, based on the WPF composite engine, supporting hardware acceleration. Here is an example of the surface rendered in Visual Studio with an extreme zoom in:



On the contratry, everything that comes out of the “textual” part is rapidly degraded with the Entity Framework Deigner. Moreover, the zoom level is limited

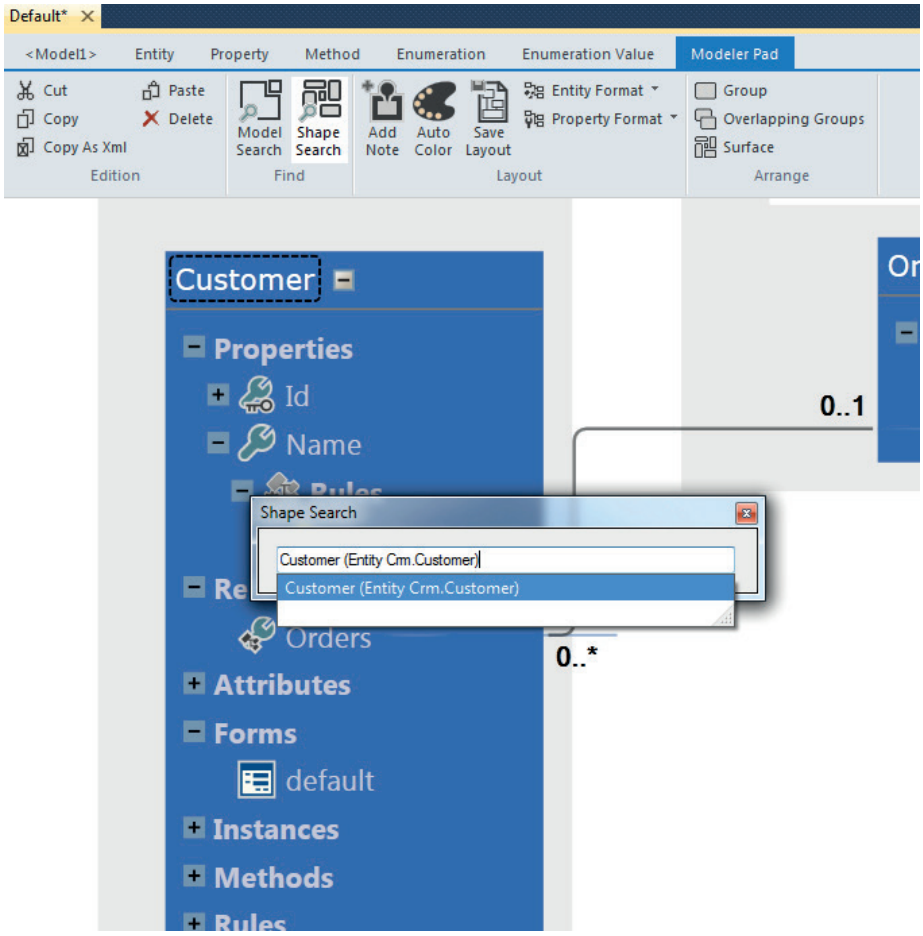


The user has also access to a menu to customize the design surface environment, using predefined styles or custom ones:



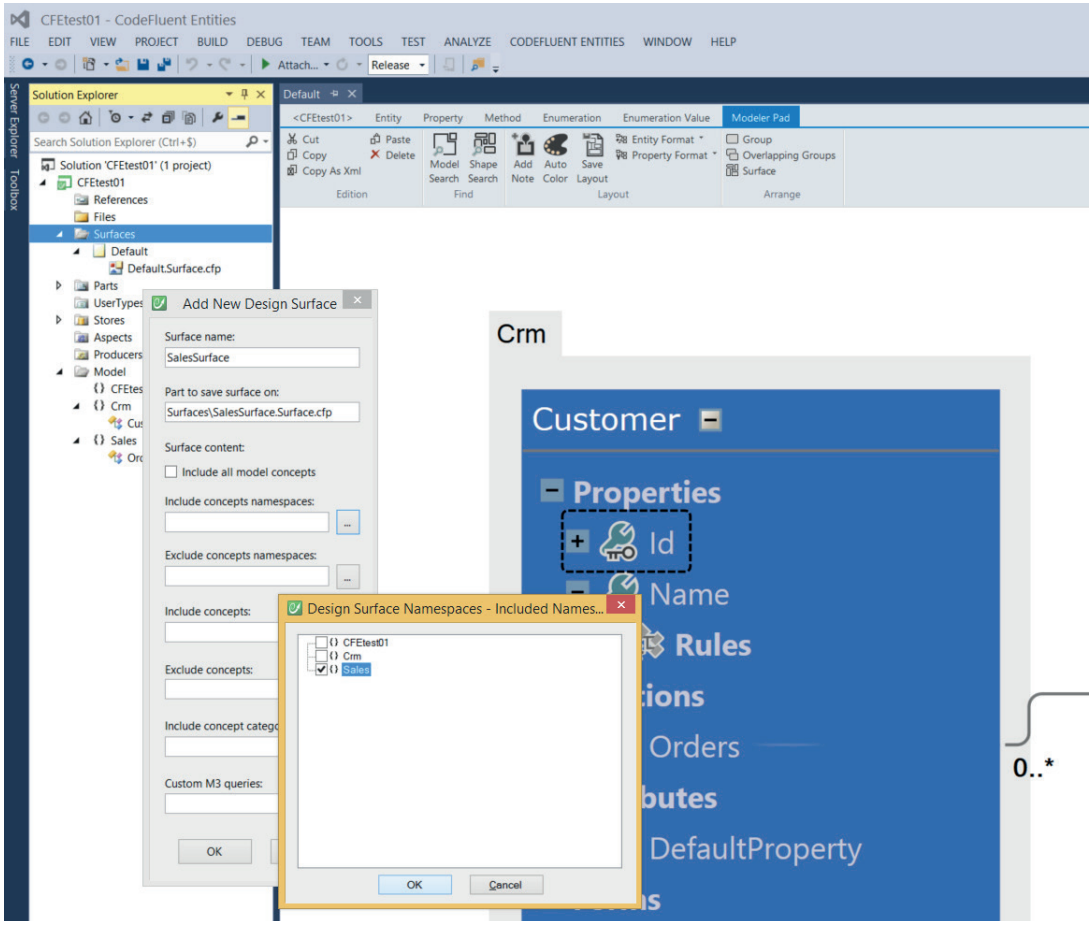
b. Form Search Engine

The **Shape Search Engine** enables a quick visual browse through hundreds of shapes:



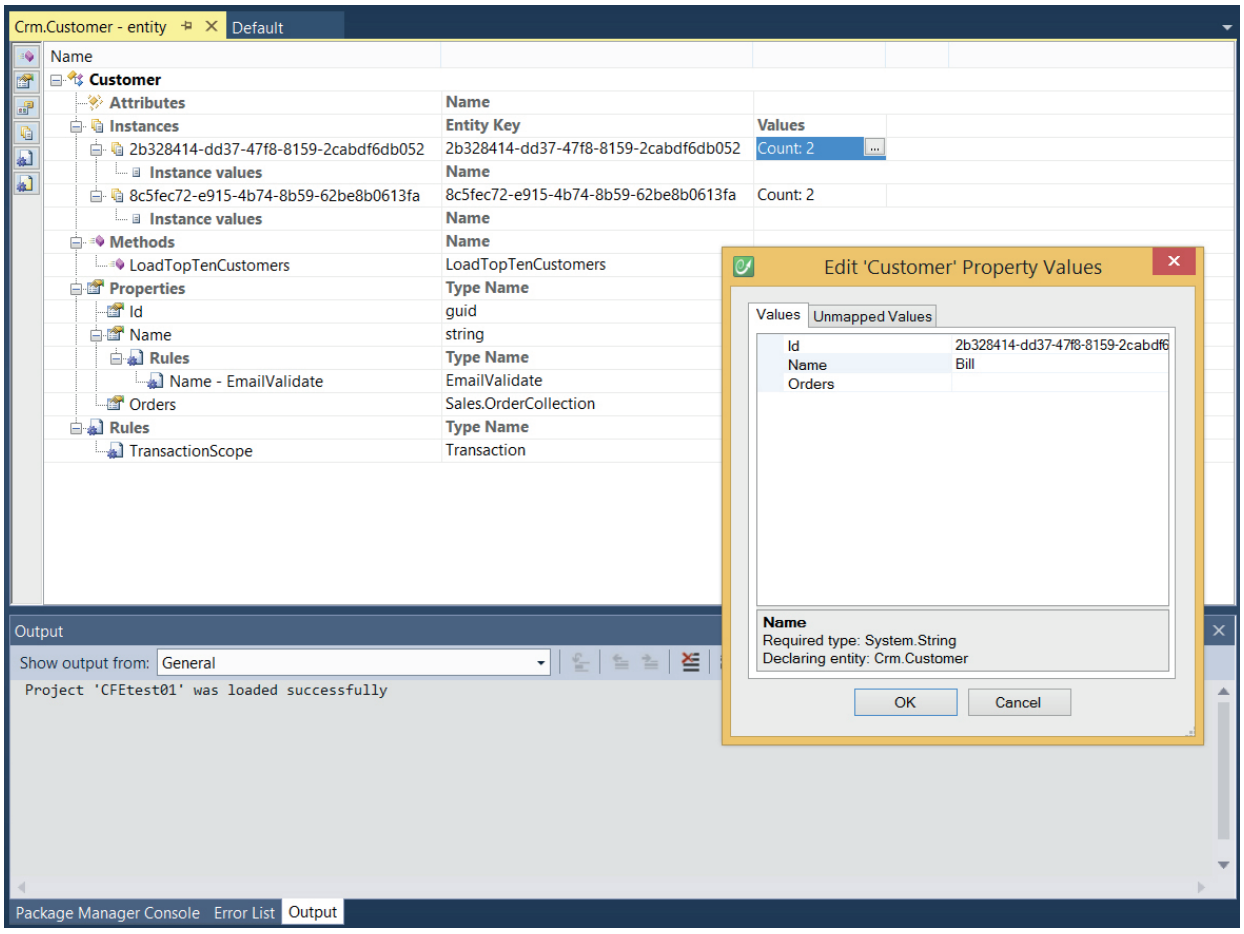
c. Multi-Surface design

The user can define any number of graphical design surfaces. Each surface is defined by a list of selectors that determine what shapes will be drawn in the surfaces. For example, in the following image, the user is defining a new surface that will display only shapes in a specific “Sales” namespace:



d. Model Grid

The Model Grid is a special exclusive editor that combines a Tree View and a List View and allows batch editing of the model, with easy keyboard navigation. Using buttons on the left (custom buttons can be added), the used can toggle what concepts are shown on not. The grid is editable using the same editors as in the standard Visual Studio Property Grid. In the following example, user is modifying the “Bill” instance of the customer entity:



2. Advanced Modeling

a. Human readable storage format

A CodeFluent Entities model is entirely stored in one or multiple XML files. The schema used is very user-friendly, and the resulting XML human-readable. It is also quite easy to modify with a standard XML editor outside of Visual Studio. Here is for example a complete model that defines two entities in different namespaces, with a few properties, including a relation (Orders <-> Customer), and two instances of the Customer entity:

```
<cf:project xmlns:cf="http://www.softfluent.com/codefluent/2005/1" xmlns:cfx="http://www.so
  <cf:entity name="Customer" namespace="Crm">
    <cf:property name="Id" key="true" />
    <cf:property name="Name" />
    <cf:property name="Orders" typeName="Sales.Order" />
    <cf:instance>
      <cf:instanceValue name="Id">3e0d0973-8660-432f-9b1e-a3e71b820536</cf:instanceValue>
      <cf:instanceValue name="Name">Bill</cf:instanceValue>
    </cf:instance>
    <cf:instance>
      <cf:instanceValue name="Id">a9db7074-c98a-4743-a874-932661c17d06</cf:instanceValue>
      <cf:instanceValue name="Name">Joe</cf:instanceValue>
    </cf:instance>
  </cf:entity>
  <cf:entity name="Order" namespace="Sales">
    <cf:property name="Id" key="true" />
    <cf:property name="Customer" typeName="Crm.CustomerCollection" />
  </cf:entity>
</cf:project>
```

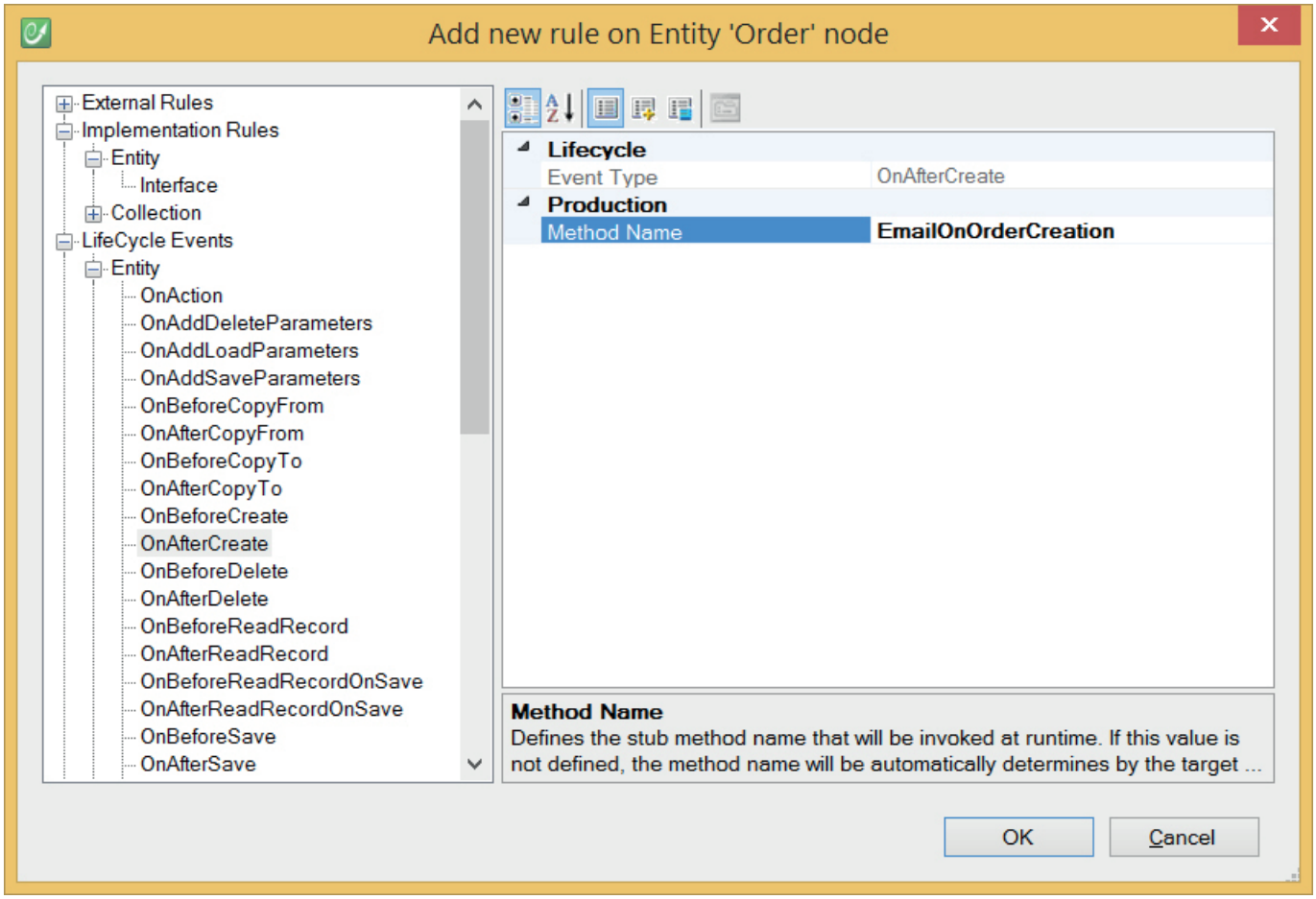
b. Multi-File storage

The storage format is XML but a CodeFluent Entities model can be stored in more than one XML file. Which concept is stored on which file is entirely up to the user. For example, one can decide to define one file per namespace, or one file per type (entity, enumeration), or any other specific layout.
This is a very useful feature that greatly helps teamwork (each team can owns files) when models are big, or in source-controlled environments (the file is the unit of work/check out/check-in).

c. Rule Editor

The **Rule** sub-concept in CodeFluent Entities models applies to the **Project, Entity, Method,** and **Property** concepts. Rules are not just property validation rules (IDataError, Data Annotations, etc.) but can also be Transaction rules, Security rules, LifeCycle rules, etc.

The visual modeler includes a complete Rule Editor that fully supports this concept:



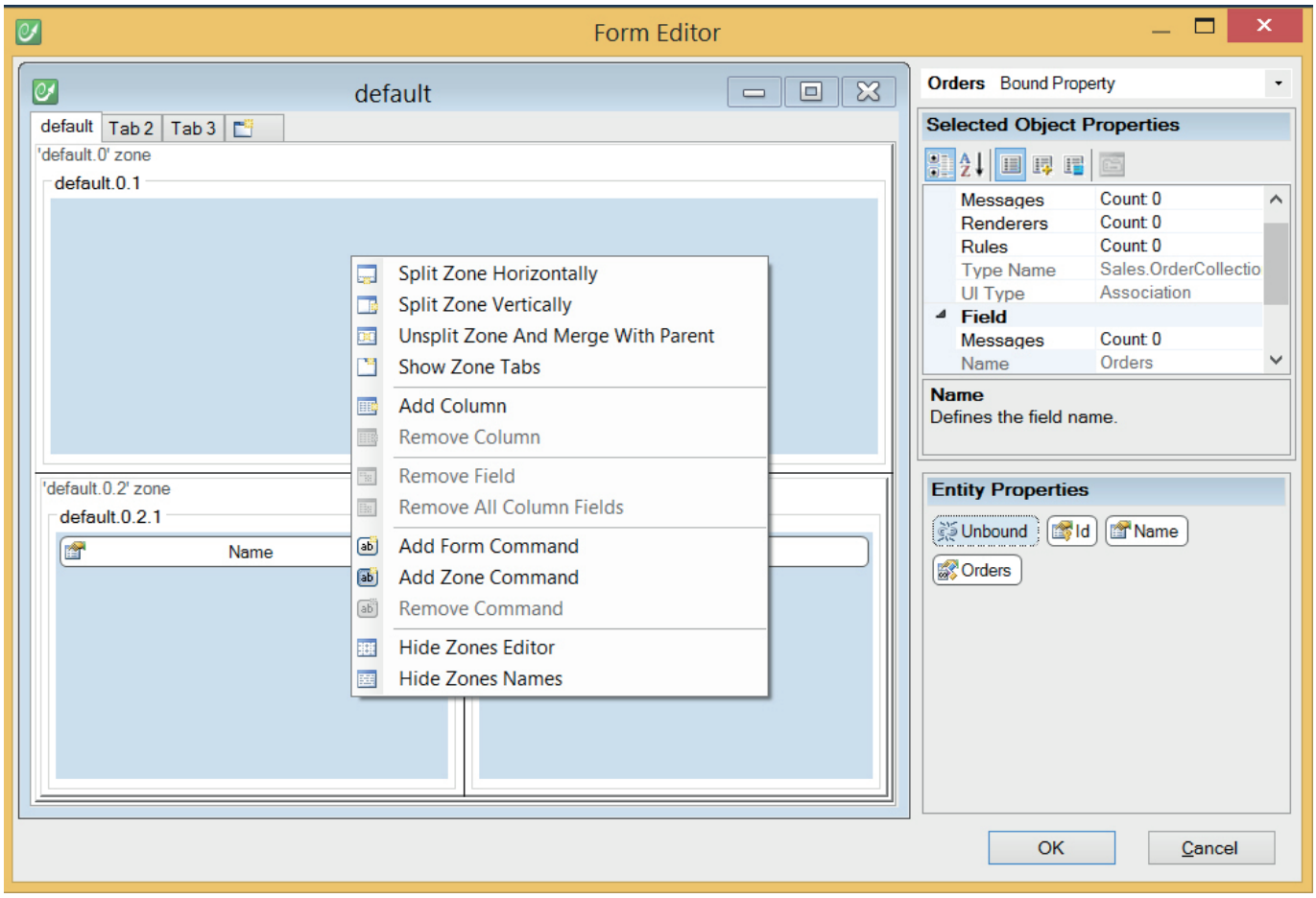
The BOM producer (the one that generates .NET classes) will ultimately use these rules to generate some extra code automatically.

d. Platform-Independent Form Editor

CodeFluent Entities defines a whole set of UI-oriented concepts: **Form, Tabs, Zone, Column, Field**, etc.

These concepts are supported by recent UI producers that ship out-of-the-box: ASP.NET Producer V2 (ships with 3 templates: Ajax/JQuery/jqGrid, ASP.NET MVC 3 and ASP.NET WebForms classic), Smart Client Producer, and SharePoint WebParts producer.

The visual modeler includes a platform-independent form editor that allows a developer to design graphically a Form attached to a given entity:



e. Aspects and Dynamic Modeling

CodeFluent Entities introduces a unique feature called “CodeFluent Aspects”. CodeFluent Aspects are .NET programs, written by the user, allowed to run during the inference phase of a given model, before the actual generation happens. They are called “aspects” as they impact the behavior of concepts involved.

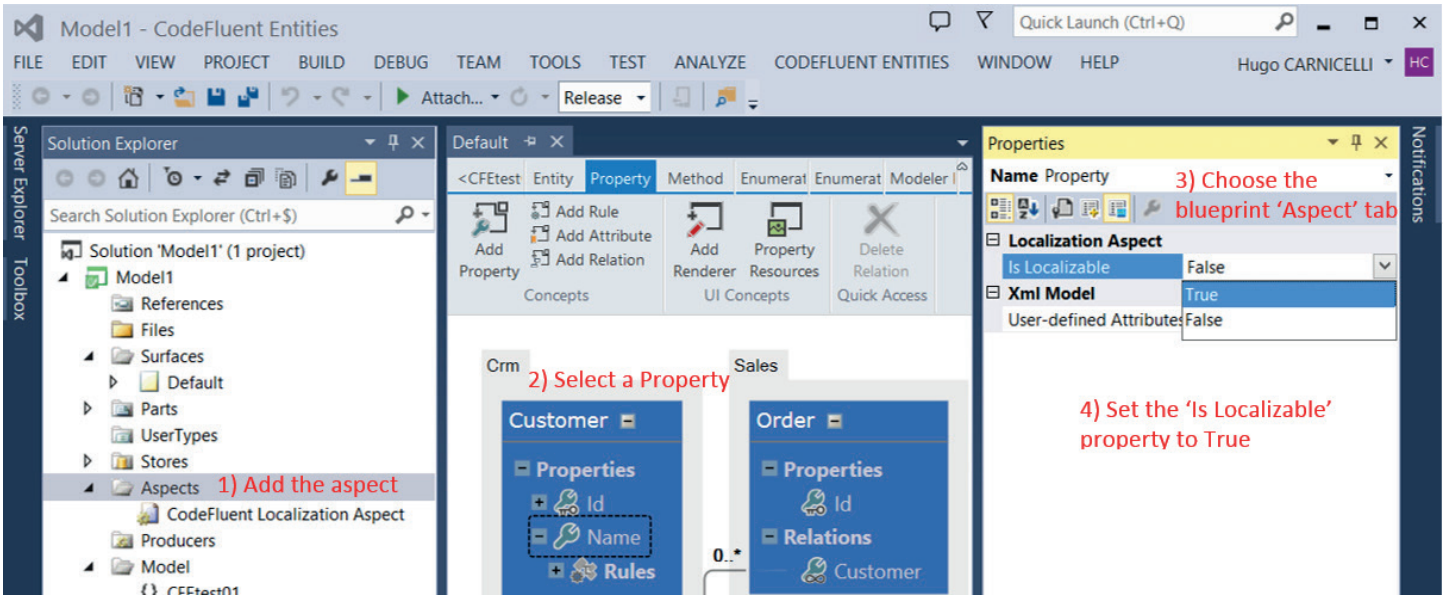
For example, if a custom aspect adds a “MyProp” property dynamically to all entities in the model, the inference engine will add automatically all the corresponding artifacts in all generated layers, such as columns to tables, parameters to stored procedures, expression in stored procedure code for Save, .NET code to generated classes, to WCF interfaces, etc., just as if the property was added manually by the user with the graphical surface.

CodeFluent Entities Aspects truly enable the concept of “Software Factory”, automating the development and maintenance of every behavior that can be mechanized in a given model. Huge models with hundreds of entities benefit heavily from this feature.

These Aspects programs (C# or VB.NET) can be stored directly in the model XML file (using raw snippet code), or can be compiled in external assemblies and in the same Visual Studio solution. They can even be shared across models, when their purpose is general and can be applied to different scenarios. Aspects can be self-described with meta-information, so their usage by the developer who has not written the aspect is 100% integrated within Visual Studio, as shown below.

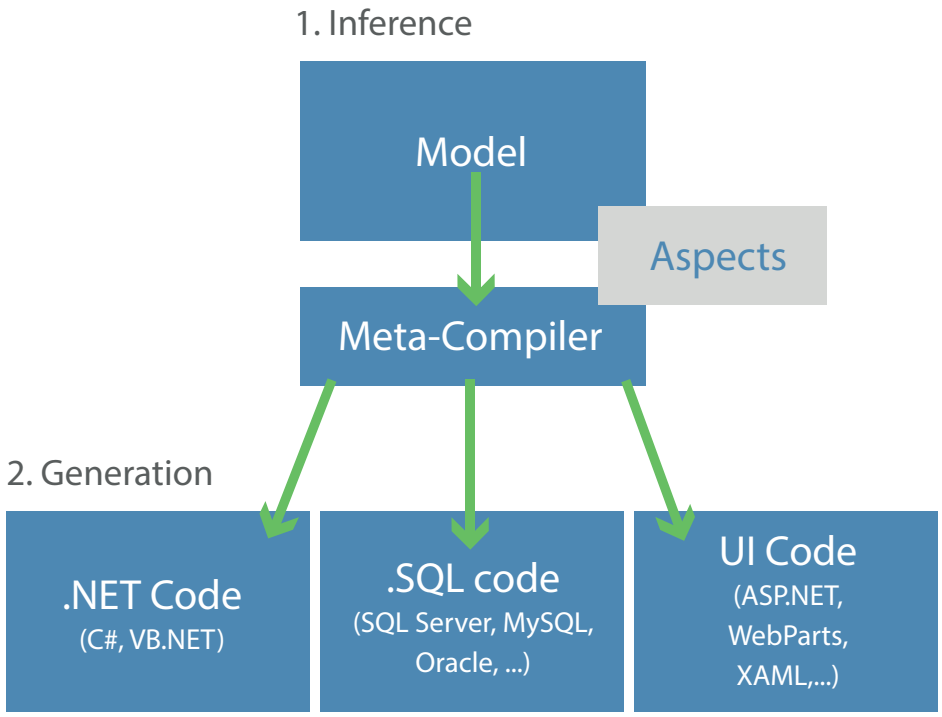
The product is shipped out-of-the-box with official aspects. One of the most used one is the “Localization” aspect which enables the localization of the database, creating tables, columns, persistent views, and stored procedures to hold the localized variants of a given property.

To enable this for a given property, a regular developer just need to add the aspect to the project, and declare which property will be localizable. The aspect will ensure everything will be modified in all layers accordingly:



3. DBA-Friendly

CodeFluent Entities is not an ORM; it's a Model-First tool. It means the database does not need to be designed first, nor the code, because the tool will in fact generate both (and other artifacts) from the model.



This being said, the generated database code (and.NET code as well) has been carefully reviewed and tuned to be the closest as possible to something a human could have very well written "manually", but it's a much less error-prone process.

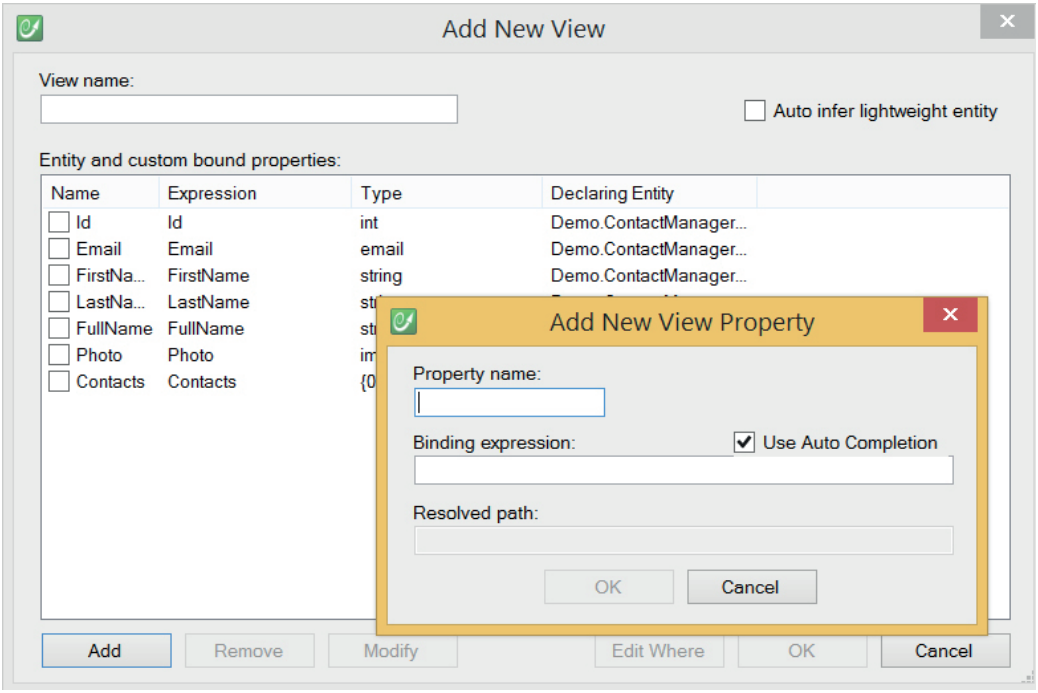
It also means the code (.NET or SQL) will be generated at design time, **not** at runtime. CodeFluent Entities does not rely on cryptic dynamic code (.NET or SQL) which you never get a chance to understand, debug, or even replace if you don't like it. Since all code is generated at design time, you will be able to have a deep look at it. Since the product also relies on stored procedures for the database access (on SQL Server, Oracle Database and MySQL), you will also be able to rewrite some of these procedures when you think you can do a better job than the machine.

a. Continuous Generation and Database Diff Engine

The SQL Server Producer is equipped with a powerful difference engine that allows you to continuously generate the SQL code from your model while preserving your existing data. Whenever possible the Diff Engine will just create new tables, add new columns, remove unneeded ones (this is an option set by default), and also take care about all declared integrity constraints.

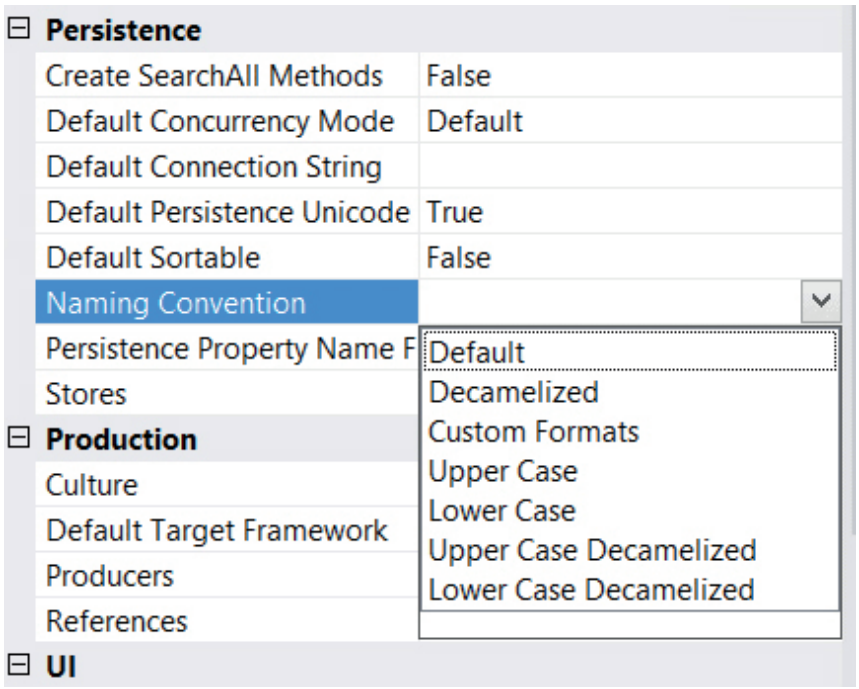
b. Persistent Views

A **View** in a CodeFluent Entities model is a quite broad concept, which can be ultimately inferred by the inference engine as a persistent view, but not only (for example, UI producers use views directly, and not persistent views). Therefore, the product comes with a View Editor:



c. Database Naming Conventions

Since CodeFluent Entities is a DBA-friendly model-first tool, it allows you to automatically choose a naming convention for the generated SQL code. By default, the product comes with standard naming conventions, but you can write your own or derive from an existing one. This option belongs to CodeFluent Entities Project properties.



V

Open challenges with Entity Framework & ORMs in general

V. Open challenges with Entity Framework & ORMs in general

Taking Entity Framework as an example, we just gathered a series of questions and discussions that demonstrate the long way ORMs still have to go through before one can really rely on them to build real-world business solutions.

“Entity Framework: Mapping sproc results to existing EF Entity”

“SQL Variant in EF”

“SQL Server 2008 data types and EF”

“Entity Framework CTP5 - How to Call Stored Procedure?”

http://stackoverflow.com/questions/21289340/how-can-i-use-ef6-to-update-a-many-to-many-table

http://stackoverflow.com/questions/4739806/entity-framework-mapping-sproc-results-to-existing-ef-entity

http://stackoverflow.com/questions/4789788/entity-framework-ctp5-how-to-call-stored-procedure

http://social.msdn.microsoft.com/Forums/eu/adonetefx/thread/0862812c-b9a8-451f-ad1d-af478a21cce0

http://www.codeproject.com/KB/database/sql_variant_in_ef.aspx

http://thedatafarm.com/blog/data-access/sql-server-2008-data-types-and-entity-framework-4/

The Entity Framework is quite limited by the types it supports. It does not totally support .NET enumerations, SQL Server User Defined Types (UDT), SQL Server “sql_variant” type, , custom .NET objects (even serializable ones. CodeFluent Entities does all this out-of-the-box.

“EF Performance: ComputeHashValue() in query compilation”

“Why is my code doing lazy loading even after I turned it off at every possible point?”

“Should the order of LINQ query clauses affect Entity Framework performance?”

“LINQ to SQL execution time is 50x longer than SSMS SQL”

http://stackoverflow.com/questions/23360660/ef-performance-computehashvalue-in-query-compilation

http://stackoverflow.com/questions/22626976/why-is-my-code-doing-lazy-loading-even-after-i-turned-it-off-at-every-possible-p

http://stackoverflow.com/questions/17323547/should-the-order-of-linq-query-clauses-affect-entity-framework-performance

http://stackoverflow.com/questions/14928939/linq-to-sql-execution-time-is-50x-longer-than-ssms-sql

With CodeFluent Entities, there is no dynamic code making the code predictable and dev-friendly to debug. The following post, http://blog.codefluententities.com/2014/03/27/fetch-performance-of-codefluent-entities/, shows how to change the generated code to fit your expectations by using aspect.

“Best option to Cascade Delete in Entity Framework”

“How cascade delete really works in EF”, “Cascading deletes on *Derived Entities*”

http://stackoverflow.com/questions/22582585/best-option-to-cascade-delete-in-entity-framework

http://blogs.msdn.com/b/alexj/archive/2009/08/19/tip-33-how-cascade-delete-really-works-in-ef.aspx

http://stackoverflow.com/questions/13461534/cascading-deletes-on-derived-entities

http://social.msdn.microsoft.com/Forums/en-US/040e7eaa-a561-4c3b-84af-851e5ec67c60/delete-does-not-cascade-to-inherited-entity-tables?forum=adodotnetentityframework

CodeFluent Entities supports cascade delete behavior for in-memory data as well as in the database.

“Entity Framework 4 - Handling very large (1000+ tables) data models?”, “Multiple Diagrams in Model vs One large .EDMX vs Multiple .EDMX”

http://stackoverflow.com/questions/2981054/entity-framework-4-handling-very-large-1000-tables-data-models

http://stackoverflow.com/questions/14956785/multiple-diagrams-in-model-vs-one-large-edmx-vs-multiple-edmx

We have customers using CodeFluent Entities (even 32-bits version) with models containing more than 600 entities.

“Entity Framework 4 Generate Database From Model With Multiple Schemas”

http://stackoverflow.com/questions/3669766/entity-framework-4-generate-database-from-model-with-multiple-schemas

The Entity Framework does not support much grouping features such as database schema or .NET namespaces. CodeFluent Entities has full supports for database schemas and .NET namespaces, and the possibility to automatically map them.

“Entity framework: Need a easy going, clean database migration solution”

“EF Code First - Recreate Database If Model Changes”

“Keep Database Content On Model Change”

http://stackoverflow.com/questions/5463121/keep-database-content-on-model-change

The Entity Framework does not allow toapply database differences between two versions of database model. CodeFluent Entities is capable of updating the SQL Server database continuously for every model update while preserving existing data (when technically possible).

“How to evade writing a lot of repetitive code when mapping?”

http://stackoverflow.com/questions/4542561/how-to-evade-writing-a-lot-of-repetitive-code-when-mapping

This is the very first reason why we created CodeFluent Entities. Whatever ORM tool you will use, you will still have to write a lot of repetitive code, or define custom templates that you will have to maintain or pick 3rd parties one that will introduce another dependency in your organization.

“OptimisticConcurrencyException Does Not Work in Entity Framework In Certain Situations”

“Entity Framework many-to-many relationship include extremely slow”

“Why does EF generate this sql when quering reference property ?“

http://stackoverflow.com/questions/4402586/optimisticconcurrencyexception-does-not-work-in-entity-framework-in-certain-situa

http://stackoverflow.com/questions/4531128/entity-framework-many-to-many-relationship-include-extremely-slow

http://stackoverflow.com/questions/4829286/why-does-ef-generate-this-sql-when-quering-reference-property

Because the Entity Framework is based on “meta” techniques and dynamically generated code, it can be sometimes quite difficult to debug and understand, like many ORM tools. CodeFluent Entities’ generated code is

```
<Employee>
	<Id />
	<Contact typeName="Contact" />
	<cf:method name="LoadCustom" body="load where contact.FirstName = 'A' or
	contact.FirstName = 'B' or contact.FirstName = 'C' " />
</Employee>
<Contact>
	<Id />
	<FirstName />
</Contact>
```

And the generated .SQL code and stored procedure would simply be

```
SELECT DISTINCT
[Employee].[Employee_Id],[Employee].[Employee_Contact_Id],[Employee].[_trackLastWri
teTime],[Employee].[_trackCreationTime],[Employee].[_trackLastWriteUser],[Employee]
.[_trackCreationUser],[Employee].[_rowVersion]
	FROM [Employee]
	Left Outer Join [Contact] On ([Employee].[Employee_Contact_Id] =
[Contact].[Contact_Id])
	WHERE (([Contact].[Contact_FirstName] = 'A') Or
(([Contact].[Contact_FirstName] = 'B') Or ([Contact].[Contact_FirstName] = 'C')))
```

“How to update entity from new created detached entity”

“Is there a way to accomplish cascading copies of Entities in EntityFramework?”

http://stackoverflow.com/questions/2358575/how-to-update-entity-from-new-created-detached-entity

http://stackoverflow.com/questions/7326489/is-there-a-way-to-accomplish-cascading-copies-of-entities-in-entityframework

CodeFluent Entities creates Clone & CopyTo methods out-of-the-box.

“Handling BLOBs in Entity Framework 4.0 in a stream-fashion”

http://stackoverflow.com/questions/4784367/handling-blobs-in-entity-framework-4-0-in-a-stream-fashion

CodeFluent Entities defines advanced types such as “file”, “image”, “video”, “audio” and “blob”. When these types are used in properties, the builder will generate sophisticated code in all layers that will leverage built-in runtime classes to bring to the developer advanced capabilities such as streaming from end-to-end (including sequential access from the database), caching, metadata (file name, file size, attributes, content-type.), thumbnails creation and automatic binding (Winforms, WPF). There is no built-in equivalent on the market from any other tool.

“How to create read-only entity in Entity Framework?”

http://stackoverflow.com/questions/4828935/how-to-create-read-only-entity-in-entity-framework

This demonstrates the Entity Framework was built from the grounds up with the word “data access” in mind, before anything else. CodeFluent Entities supports non persistent entities.

“How does EF4 compare with NHibernate?”

http://stackoverflow.com/questions/4847210/how-does-ef4-compare-with-nhibernate

This thread details some of the Entity Framework issues. CodeFluent Entities does not have any of these issues.

- It has full support for enumerations. You can model and generate an enumeration or reuse an existing one from an already compiled assembly.
- It defines the notion of naming convention, and ships with a few ones. You can develop your own.
- It generates human-readable.
- It has batch delete support.
- It supports model evolution and continuous building.
- It has many extensibility points at all levels.

“Seemingly infinite stack trace in EF 4.0 and poor query performance under load”

“Linq to SQL throwing a StackOverflowException”

http://stackoverflow.com/questions/6978207/seemingly-infinite-stack-trace-in-ef-4-0-and-poor-query-performance-under-load

http://stackoverflow.com/questions/5744764/linq-to-sql-throwing-a-stackoverflowexception

Those threads illustrate how hard it can be to debug dynamically creating data access code. This is why CodeFluent Entities generates predictable, human readable stored procedures called by corresponding .NET methods.

“How to store images using Entity Framework Code First CTP 5?”

“Save a file in SQL Server 2008 database with Entity Framework”

http://stackoverflow.com/questions/4653095/how-to-store-images-using-entity-framework-code-first-ctp-5

http://stackoverflow.com/questions/5370351/save-a-file-in-sql-server-2008-database-with-entity-framework

Blobs (e.g. images, videos, documents, audios) are a native type in CodeFluent Entities, handling blob properties across layers is seamless. Furthermore, blobs are manipulated across all layers of your application (and note byte tables).

“How can I add constraints to an ADO.NET Entity?”

http://stackoverflow.com/questions/1963829/how-can-i-add-constraints-to-an-ado-net-entity

CodeFluent Entities supports this feature since day one.

“EF with Azure - Mixing SQL Server and Windows Azure Storage”

http://stackoverflow.com/questions/7580649/ef-with-azure-mixing-sql-server-and-windows-azure-storage

Another feature provided out-of-the-box by CodeFluent Entities and which does not require a single line of code from the developer. Developers can chose to store their blobs in database (by default), in the Windows Azure Blob Storage, or in a NTFS directory on a server.

“Add XML documentation / comments to properties/fields in EF generated classes”

http://stackoverflow.com/questions/7672627/add-xml-documentation-comments-to-properties-fields-in-ef-generated-classes

Using CodeFluent Entities you can document the generated code without having to neither use another tool nor create custom templates.

“How to make EntityFramework add rows in order?”

http://stackoverflow.com/questions/5159824/how-to-make-entityframework-add-rows-in-order

Saving auto-incrementing fields is fully supported in CodeFluent Entities.

“How should I annotate CreatedOn and ModifiedOn columns with EF 4.1?”

Tracking columns are (by default) automatically generated and fully supported in CodeFluent Entities.

“Internationalization of content in Entity Framework”

http://stackoverflow.com/questions/11085205/internationalization-of-content-in-entity-framework

The “localization aspect” is provided out-of-the-box and allows the creation data localization columns in the database without any extra work

“Autogenerated timestamp in Entity Manager”

http://stackoverflow.com/questions/6087083/autogenerated-timestamp-in-entity-manager

CodeFluent Entities generates automatically a timestamp column (“_rowversio”n) which is available on all platforms (e.g. Oracle, SQL Server, SQL Azure) and which does not require a single line of custom code.

“Dealing with DateTime using “Linq Methods”

http://stackoverflow.com/questions/6253905/dealing-with-datetime-using-linq-methods

In CodeFluent Entities, database values are automatically converted into model object values and vice versa and all those conversions use default values so developers don’t ever have to struggle with nulls or DbNulls for value types.

“LINQ - Querying about 6000 unique records by WHERE clause”

http://stackoverflow.com/questions/8730879/linq-querying-about-6000-unique-records-by-where-clause

Those kind of queries are supported out-of-the-box by CodeFluent Entities and do not require any custom code.

“LINQ to Entities does not recognize the method ‘Double Parse(System.String)’ method, and this method cannot be translated into a store expression”

“Problem of cascade delete in entity framework”

http://stackoverflow.com/questions/5971521/linq-to-entities-does-not-recognize-the-method-double-parsesystem-string-met

http://stackoverflow.com/questions/6297657/ef4-1-fluent-api-sqlquery-configuration-mappings-when-calling-sproc-data

http://stackoverflow.com/questions/6443917/problem-of-cascade-delete-in-entity-framework

Those threads illustrate how ORMs tend to not be DBA-friendly. CodeFluent Entities on the other hand is built to be very DBA-friendly and for instance, those topics are all handled natively and without the need of any custom

http://stackoverflow.com/questions/7073147/entity-framework-t4-poco-objects-raising-exception-in-wcf

Using CodeFluent Entities, the generated .NET classes can be used across all platforms (e.g. Windows Forms, WPF, ASP.NET Web Forms, ASP.NET MVC, WCF, WF, Silverlight, and SharePoint).

“Upgraded to the latest EF (4.1) from CTP - how do we override underscores generated in foreign keys”

http://stackoverflow.com/questions/7487746/upgraded-to-the-latest-ef-4-1-from-ctp-how-do-we-override-under-scores-genera

CodeFluent Entities allows developers to implement custom naming conventions which enable them to change how all persistence objects (e.g. tables, stored procedures, columns, views, and constraints) are named.

http://stackoverflow.com/questions/7073147/entity-framework-t4-poco-objects-raising-exception-in-wcf

Using CodeFluent Entities, the generated .NET classes can be used across all platforms (e.g. Windows Forms, WPF, ASP.NET Web Forms, ASP.NET MVC, WCF, WF, Silverlight, and SharePoint).

“Upgraded to the latest EF (4.1) from CTP - how do we override underscores generated in foreign keys”

http://stackoverflow.com/questions/7487746/upgraded-to-the-latest-ef-4-1-from-ctp-how-do-we-override-under-scores-genera

CodeFluent Entities allows developers to implement custom naming conventions which enable them to change how all persistence objects (e.g. tables, stored procedures, columns, views, and constraints) are named.

VI

Resources

VI. Resources

Product Home Page:
http://www.softfluent.com/products/codefluent-entities

Get a trial version (full-featured and free for personal usage):
http://www.softfluent.com/trials/codefluent-entities/

Why CodeFluent Entities (with samples)?
http://www.softfluent.com/products/codefluent-entities/why/examples

Get Started with CodeFluent Entities
http://www.softfluent.com/products/codefluent-entities/get-started

CodeFluent Entities Blog (more than 350 articles!):
http://blog.codefluententities.com/

Forums:
http://forums.softfluent.com/

Reference Documentation:
http://www.softfluent.com/community-support/codefluent-entities/documentation

CodeFluent Entities on StackOverflow:
http://stackoverflow.com/search?q=codefluent+entities

www.softfluent.com
info@softfluent.com

EUROPE
3 rue de la Renaissance
92160 Antony
France

AMERICA
2018 156th Avenue NE
Bellevue, WA 98007
USA