GENESYSSOURCE

How-to Pull Your Data through the Customer

A Cross-Platform Framework Quick-Start

GENESYSSOURCE
22431 Antonio, Suite B160-843
Rancho Santa Margarita, Ca. 92688
+1.949.544.1900 | GenesysSource.com

Created: Feb 9, 2017
Modified: September 24, 2018

Contents

Introduction

Genesys Source Framework is an open-source C# business object reusability framework…enabling your objects to be cross-platform, full-stack, CRUD-to-SQL/Services enabled, self-serializing, self-validating and self-tracking.

Your business object DLLs are then consumable by your app, using any .NET web or native technology, including MVC, Web API, UWP, Xamarin iOS, Xamarin Android, Web Forms, WCF, Windows Service, Console Apps and CLR stored procedures.

The goal is enabling developers to reuse business objects, painlessly…without the plumbing time and cost.

Features of the Genesys Framework:

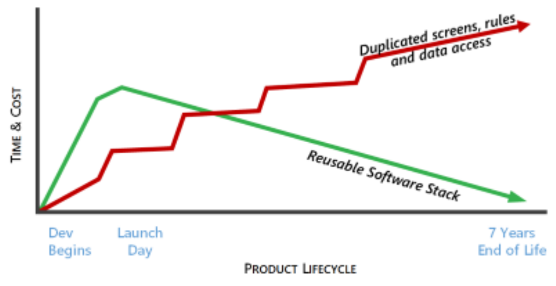
- *CrudEntity* base with CRUD-to-SQL data access, *Create()*, *Read(Expression)*, *Update()* and *Delete()*.
- *ViewModel (MVVM) and MvcController (MVC)* base with CRUD-to-Services data access, *CreateAsync()*, *ReadAsync(Expression)*, *UpdateAsync()* and *DeleteAsync()*.
- *ModelEntity* base with *Serialize*() for binding to screens in MVC, WPF, UWP, etc.
- *SaveableDatabase* and *ReadOnlyDatabase* for direct data access via EF -> SQL.
- *Genesys.Extensions*: Cross-platform common library.

Why reusable business objects?

Code reuse is an important theme in many of todays accepted software practices, such as N-tier and Object-oriented programming (OOP.)

Typically, reusable software stacks and services have low technical debt and are cheaper to maintain over time. Reusable code "settles" over time and costs decrease. Your return on investment (RoI) is greater with reusable software stacks

Conversely, the code duplication method tends to cost more over time, with high technical debt in the form of maintenance time and costs spiking per each duplicated item. Your costs go up over time, until the software is rewritten or retired.



The Genesys Framework offers n-tier, reusable business objects, with a low learning curve. Reusability without the cost of doing it yourself, and without the uncertainty of an untested new code base.

Why cross-platform, full-stack business objects?

Microsoft .NET classes have a unique characteristic…they can run almost anywhere on any popular platform and run in any software tier. This allows a .NET entity class, like a Customer entity, to enjoy a 100% strongly-typed stack and consistency in properties and validation rules…in web sites, web services, native apps, CLR stored procedures and in class libraries.

With cross-platform full-stack entity objects, spelling errors and type errors show immediately as a compile error…in a stored procedure, in a data access C# file, in a MVC controller…everywhere that entity is used. Typing is maintained through the stack:

Any SQL Data -> Framework.Database -> Framework.DataAccess -> Framework.Models -> Any .NET App

Genesys Framework takes advantage of run-anywhere to enable any business object to run in Web, Services, Desktop and Mobile.

Take the Customer entity as an example:

- *CustomerInfo.cs:* Heavy Data Access Object (DAO) based on Entity Framework database-first. Supports CRUD-to-SQL methods of *Create*(), *Read*(*Expression*), *Update*(), *Delete*().
- *CustomerModel.cs*: Lightweight screen and transport models. This class is cross-platform and runs in MVC, Web API, UWP, WPF, Xamarin iOS, Xamarin Android, CLR Stored Procedures.
- *CrudViewModel<CustomerModel>*: MVVM ViewModel with CRUD-to-Services methods such as *CreateAsync*(), *ReadAsync*(*Expression*), *UpdateAsync*() and *DeleteAsync*().
- *customer.Serialize()*: JSON string is returned from any class that inherits *CrudEntity* or *ModelEntity*. This JSON can be controller generated and used by client-side web applications.

Pre-requisites

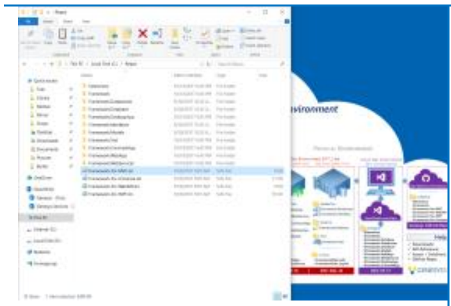
To get the most out of this article, the following skills are recommended:

- Moderate C# and .NET, HTML and XAML
- Low-Moderate T-SQL and Database design
- Awareness of N-tier, MVC, MVVM and REST
- Visual Studio Community (or greater) from https://www.visualstudio.com/downloads/
- SQL Server Management Studio from https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms

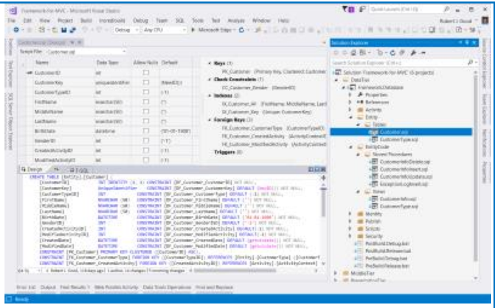
Add a new Field/Property to the Customer object

This procedure walks you through the process of adding, changing or deleting an entity field. Including the column in the database, the data access object, the model and a MVC View.

①	②	③
Open	Open Table & Add Column	Open View & Add Column
Framework-for-MVC.sln	*Tables\Customer\Customer.sql*	*Views\CustomerCode\CustomerInfo.sql*

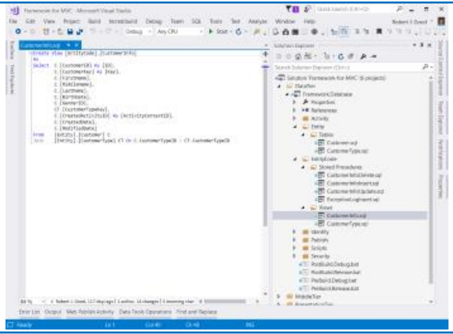


1. Open the *Framework-for-MVC.sln* Visual Studio solution file
   - Default: C:\Source\Framework-for-MVC.sln



2. Navigate to and open Customer table:
   Framework.Database\Tables\CustomerCode\CustomerInfo.sql
3. Add a new field: NickName

```
[NickName] NVARCHAR (50) CONSTRAINT
[DF_Customer_NickName] DEFAULT ('') NOT NULL,
```



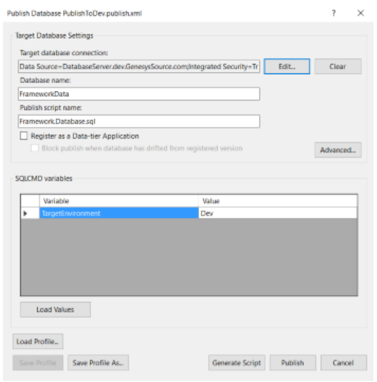
4. Navigate to and open Customer view:
   Framework.Database\Views\CustomerCode\CustomerInfo.sql
5. Add the NickName field:

```
C.[NickName],
```

4 Publish *FrameworkData* to SQL Server

5 Open *CustomerInfo.cs in Framework.DataAccess*

6 Add NickName to *CustomerInfo*



Publish Database PublishToDev.publish.xml

Target Database Settings

Target database connection: Data Source=DatabaseServer.dev.GenesysSource.com;Integrated Security=Tr

Database name: FrameworkData

Publish script name: Framework.Database.sql

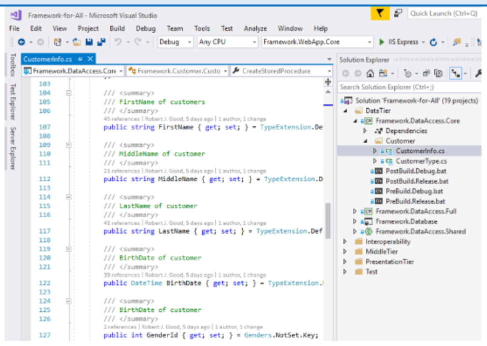
Register as a Data-tier Application

Block publish when database has drifted from registered version

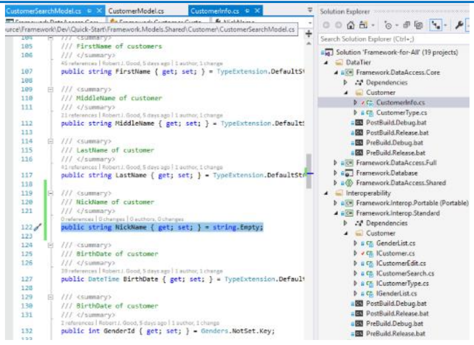
SQLCMD variables

Variable	Value
TargetEnvironment	Dev

6. Open SSDT publish screen:
   Framework.Database\Publish\PublishToDev.publish.xml
   - Ensure *Target database connection* is correct
7. Click *Generate Script* and review
8. Click *Publish* to push changes to SQL



9. Open *CustomerInfo.cs*
   - *Framework.DataAccess\Customer*



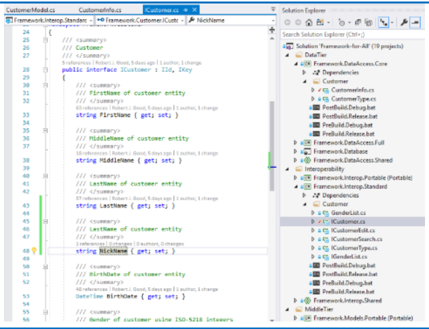
10. Add NickName by Copy/Paste the following property:

```
public string NickName { get; set; } =
string.Empty;
```

7 Add NickName to ICustomer & Models

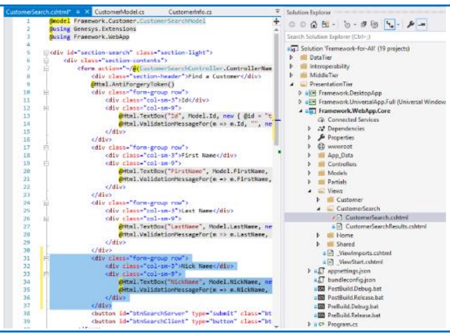
8 Add NickName to Framework.WebApp Search

9 Run!



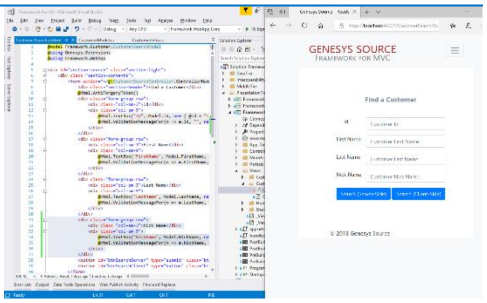
```
{
    /// <summary>
    /// Customer
    /// </summary>
    public interface ICustomer : IId, IKey
    {
        /// <summary>
        /// FirstName of customer entity
        /// </summary>
        string FirstName { get; set; }

        /// <summary>
        /// MiddleName of customer entity
        /// </summary>
        string MiddleName { get; set; }

        /// <summary>
        /// LastName of customer entity
        /// </summary>
        string LastName { get; set; }

        /// <summary>
        /// LastName of customer entity
        /// </summary>
        string NickName { get; set; }

        /// <summary>
        /// BirthDate of customer entity
        /// </summary>
        DateTime BirthDate { get; set; }

        /// <summary>
        /// Gender of customer using ISO-5218 integers
```

11. Open Framework.Interfaces\Customer \ICustomer.cs
12. Add NickName property as a string - Notice all classes that implement ICustomer throw an error requiring ICustomer.NickName
13. Add NickName to all dependent models (CustomerModel and CustomerSearchModel)



```
@model Framework.Customer.CustomerSearchModel
@using Genesys.Extensions
@using Framework.WebApp

<div id="section-search" class="section-light">
    <div class="section-contents">
        <form action="~/@(CustomerSearchController.ControllerNam
            <div class="section-header">Find a Customer</div>
            @Html.AntiForgeryToken()
            <div class="form-group row">
                <div class="col-sm-3">Id</div>
                <div class="col-sm-9">
                    @Html.TextBox("Id", Model.Id, new { @id = "t
                    @Html.ValidationMessageFor(m => m.Id, "", ne
                </div>
            </div>
            <div class="form-group row">
                <div class="col-sm-3">First Name</div>
                <div class="col-sm-9">
                    @Html.TextBox("FirstName", Model.FirstName,
                    @Html.ValidationMessageFor(m => m.FirstName,
                </div>
            </div>
            <div class="form-group row">
                <div class="col-sm-3">Last Name</div>
                <div class="col-sm-9">
                    @Html.TextBox("LastName", Model.LastName, ne
                    @Html.ValidationMessageFor(m => m.LastName,
                </div>
            </div>
            <div class="form-group row">
                <div class="col-sm-3">Nick Name</div>
                <div class="col-sm-9">
                    @Html.TextBox("NickName", Model.NickName, ne
                    @Html.ValidationMessageFor(m => m.NickName,
                </div>
            </div>
            <button id="btnSearchServer" type="submit" class="bt
            <button id="btnSearchClient" type="button" class="bt
```

14. Open Framework.WebApp \Views\CustomerSearch \CustomerSearchResults.cshtml
15. Add Nick Name to table header and body



16. Double-check the connection string, to make sure it is pointing to the proper database
17. Right-click *Framework.WebApp* project -> click *Set as StartUp Project*
18. Press *F5* or ▶ to run

1: Adding NickName to Customer business object