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

Copyright © Genesys Source. All rights reserved. Printed in USA.

Genesys Source is a registered trademark of Genesys Source. Genesys Framework and Genesys Cloud Dev Environment are trademarks of Genesys Source.

Microsoft, Azure, HoloLens, Hyper-V, Visual Studio, Windows and Xamarin are registered trademarks of Microsoft Corporation.

All other product names and logos are trademarks and service marks of their respective companies.

This document is provided "as-is." Information in this document, including URL and other Internet website references, may change without notice. Genesys Source assumes no liability for damages incurred directly or indirectly from errors, omissions, or discrepancies between the product and this document.

An attempt has been made to state all allowable values where applicable throughout this document. Any values or parameters used beyond those stated may have unpredictable results.

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:

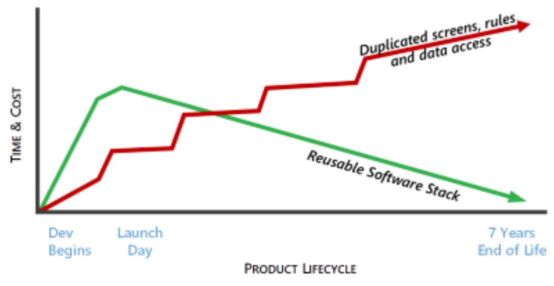
- *CrudEntity* base with CRUD-to-SQL data access, *Create()*, *Read(Expression)*, *Update()* and *Delete()*.
- *ViewModel (MVVM) and MvcController (MVC)* base with CRUD-to-Services data access, *CreateAsync()*, *ReadAsync(Expression)*, *UpdateAsync()* and *DeleteAsync()*.
- *ModelEntity* base with *Serialize*() 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

Re-wire Framework.Database SQL Views to connect to your SQL Tables

This procedure guides you through the process of re-wiring *Framework.Database.CustomerCode.CustomerInfo* view to pull data from your "Person" table. This is an example of a one-to-one swap:

1. Edit the *CustomerInfo* view.
2. De-reference the *FrameworkData Customer* table.
3. Reference your "*Person*" table that contains *First Name* or *Last Name*.
4. Alias all mismatched or missing fields in the view

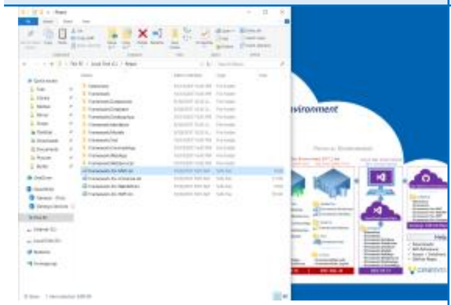
The *Genesys Framework* will now pull data via the *CustomerInfo* view, from your "Person" table

Important Tip: *For this example, keep the field names the same and column type the same (use AS keyword.) No code changes will be necessary. The existing Framework projects will work against your "Person" table as if pulling from the FrameworkData's Customer table.*

①	②	③

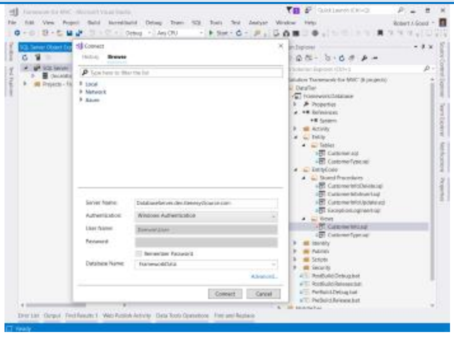
Open your Solution

i.e. Framework-for-MVC.sln



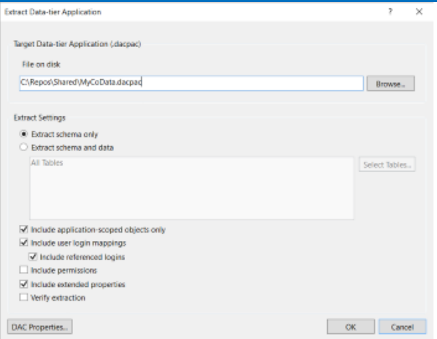
1. Open the *Framework-for-MVC.sln* Visual Studio solution file
   - Default: C:\Source\Framework-for-MVC.sln

Connect to your database in SQL Object Explorer



2. Click *View* -> *SQL Server Object Explorer*
3. Enter connection info to your database
4. Click *Connect* to add the connection

Extract your database schema to a .dacpac file



Extract Data-tier Application

Target Data-tier Application (.dacpac)

File on disk: C:\Repos\Shared\MyCoData.dacpac

Extract Settings

- [✓ Extract schema only]
- Extract schema and data

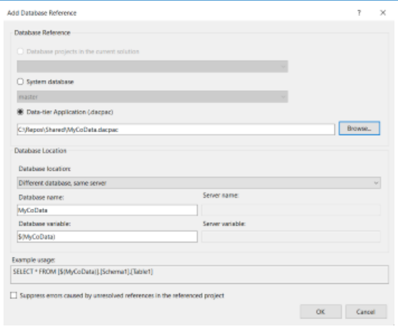
All Tables

- [✓] Include application-scoped objects only
- [✓] Include user login mappings
  - [✓] Include referenced logins
- [ ] Include permissions
- [✓] Include extended properties
- [ ] Verify extraction

5. In SQL Server Object Explorer, right-click your database -> click *Extract Data-tier Application*
6. Select: *Extract Schema Only*
7. Enter file-on-disk as:
   C:\Source\Shared\MyCoData.dacpac
8. Click *OK* to extract your schema to .dacpac

4

Add a Database Reference to your .dacpac



Add Database Reference

Database Reference

- Database projects in the current solution
- System database: master
- [✓ Data-tier Application (.dacpac)]: C:\Repos\Shared\MyCoData.dacpac

Database Location

Database location: Different database, same server

Database name: MyCoData

Server name:

Database variable: $(MyCoData)

Server variable:

Example usage: SELECT * FROM [$(MyCoData)].[Schema1].[Table1]

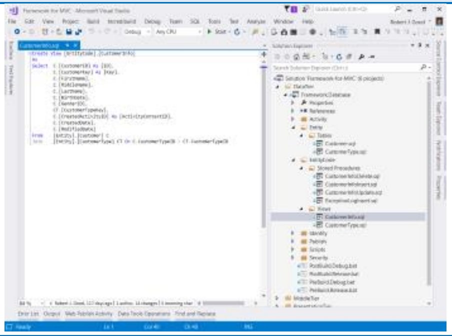
- [ ] Suppress errors caused by unresolved references in the referenced project

9. In Solution Explorer, right-click your Framework.Database\Views\CustomerCode\CustomerInfo.sql

5

Open View

Views\CustomerCode\CustomerInfo.sql



10. Navigate to and open Customer view: Framework.Database\Views\CustomerCode\CustomerInfo.sql

6

Replace the SELECT with T-SQL that pulls data from your table

For example...

If your table is: *[MyCoData].[dbo].[Cust]*

With fields: *Cust_ID, F_Name, L_Name, B_Date*

Change the SELECT to your [Cust] table...

```
Create View [CustomerCode].[CustomerInfo] As
Select     C.[Cust_ID] As [ID],
 C.[F_Name] As [FirstName],
 C.[L_Name] As [LastName],
 C.[B_Date] As [BirthDate],
... (Alias Missing Fields Here) ...
From       [MyCoData].[dbo].[Cust] C
```

11. In CustomerInfo.cs, change the SELECT statement to pull data from your database

Note: Databases must be in same SQL instance

7

Alias any missing fields with Default Values

8

Publish *FrameworkData* to SQL Server

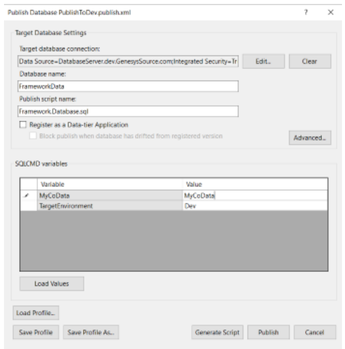
9

Run *Framework.WebApp* to pull your customer data

Alias missing fields for [Cust] example…

```
'' As [MiddleName],
-1 As [GenderID],
-1 As [ActivityContextID],
'00000000-0000-0000-0000-000000000000' As
    [Key],
'00000000-0000-0000-0000-000000000000' As
    [CustomerTypeKey],
'01/01/1900' As [CreatedDate],
'01/01/1900' As [ModifiedDate]
```

12. Alias all fields that do not have an equivalent in your customer data:
- Integer: `-1`
- String: `''`
- Date: `'01/01/1900'`
- Guid: `'00000000-0000-0000-0000-000000000000'`



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

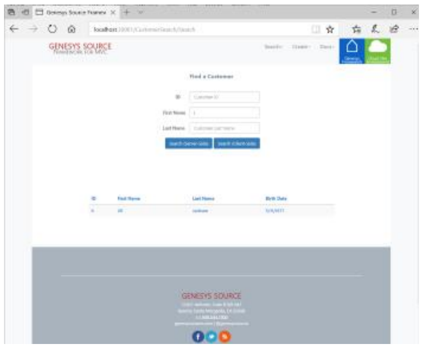
SQLCMD variables

Variable	Value
MyCoData	MyCoData
TargetEnvironment	Dev

13. Open SSDT publish screen:

Framework.Database\Publish\PublishToDev.publish.xml
- Ensure *Target database connection* is correct
- Ensure MyCoData is set to the name of your database

14. Click *Generate Script* and review
15. Click *Publish* to push changes to SQL



16. Ensure connection string is correct:

Framework.WebApp\App_Data\ConnectionStrings.json

17. Right-click *Framework.WebApp* -> click *Set as Startup Project*
18. Press *F5* or ▶ to run
    - Should run this Url: http://localhost:30001/

Search screen & customer object now pulls your data

1: Pulling your data through the CustomerInfo object

Getting Help

Have a question? Have a problem? Contact us anytime…

Contact Genesys Source	Help and Guidance	On Social
GENESYSSOURCE 22431 Antonio, Suite B160-843 Rancho Santa Margarita, CA 92688 +1 949.544.1900 www.genesyssource.com *help@genesyssource.com*	[Docs] docs.genesyssource.com [FAQs] Frequently Asked Questions [+/-] Report an Issue [Zip] Download Genesys Framework [Azure] Try Cloud Web Environment	http://twitter.com/GenesysSource http://facebook.com/GenesysSource http://GenesysSource.com/blog http://GenesysSource.com/news/rss/1