Genesys Source Framework
eBook
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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-requisites</td>
<td>5</td>
</tr>
<tr>
<td>Installing the Genesys Framework</td>
<td>5</td>
</tr>
<tr>
<td>Uninstalling the Genesys Framework</td>
<td>7</td>
</tr>
<tr>
<td>What is in the Genesys Framework?</td>
<td>8</td>
</tr>
<tr>
<td>Running the Framework.Test Integration Tests</td>
<td>9</td>
</tr>
<tr>
<td>Debugging the Genesys Framework</td>
<td>10</td>
</tr>
<tr>
<td>- Debugging Framework.WebApp (MVC) and Framework.WebServices (Web API)</td>
<td>10</td>
</tr>
<tr>
<td>- Debugging Framework.UniversalApp (UWP) and Framework.DesktopApp (WPF)</td>
<td>11</td>
</tr>
<tr>
<td>The Framework.Database project and FrameworkData database</td>
<td>12</td>
</tr>
<tr>
<td>About Framework.Database (SSDT)</td>
<td>13</td>
</tr>
<tr>
<td>Re-wire Framework.Database SQL Views to connect to your SQL Tables</td>
<td>13</td>
</tr>
<tr>
<td>Add a new Field/Property to the Customer object</td>
<td>15</td>
</tr>
<tr>
<td>Publishing the Genesys Framework to IIS and SQL Server</td>
<td>16</td>
</tr>
<tr>
<td>- Publishing Framework.Database (SSDT) to a SQL Server</td>
<td>17</td>
</tr>
<tr>
<td>- Publishing Framework.WebApp (MVC) to an IIS Web Server</td>
<td>18</td>
</tr>
<tr>
<td>- Publishing Framework.WebServices (Web API) to an IIS Web Server</td>
<td>19</td>
</tr>
<tr>
<td>Tech and Code Aspects of the Genesys Framework</td>
<td>19</td>
</tr>
<tr>
<td>- Database Connections in App_Data</td>
<td>19</td>
</tr>
<tr>
<td>- Web Service Connections in App_Data</td>
<td>20</td>
</tr>
<tr>
<td>- Framework.DataAccess pulls data through Framework.Database SQL Views</td>
<td>20</td>
</tr>
<tr>
<td>Why the Genesys Framework?</td>
<td>21</td>
</tr>
<tr>
<td>- Why build reusable code?</td>
<td>21</td>
</tr>
<tr>
<td>- Why code full-stack, cross-platform business objects?</td>
<td>21</td>
</tr>
<tr>
<td>Getting Help</td>
<td>22</td>
</tr>
</tbody>
</table>
One Framework – Your Data – Any Platform

The *Genesys Framework* is a C# framework that enables your business objects to run cross-platform, full-stack on .NET Core and .NET Framework, in Web or Mobile or Desktop...in minutes.

Tell me a little more about the *Genesys Framework*

<table>
<thead>
<tr>
<th>What is it?</th>
<th>The <em>Genesys Framework</em> is a full-stack business object framework, exposing your data as C# objects, from your SQL Database to any type of .NET app. Your <em>Customer</em> object, for example, can exist as a:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data object:</td>
<td>EF-enabled to pull data from your SQL Server via common repository-pattern methods like: <code>GetById()</code>, <code>GetAll()</code>, <code>Save()</code> and <code>Delete()</code>.</td>
</tr>
<tr>
<td>Domain object:</td>
<td>Enriched domain object containing domain behavior such as <code>IsEmployee</code>, <code>IsActive</code>, <code>HasRegistered</code>, <code>CurrentStatus</code>, etc.</td>
</tr>
<tr>
<td>View Model object:</td>
<td>Model for your screen Views that are thin, atomic and transportable. Can be extended to have view-specific properties such as a Gender select list, without altering your <code>CustomerModel</code> class.</td>
</tr>
<tr>
<td>Data-transfer object:</td>
<td>Expose your object as a DTO for data-transfer specific operations such as inter-web-services data sharing or returning your object through public API endpoints.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Why do I care?</th>
<th><strong>Full-stack projects in seconds:</strong> Your entire C# stack solution is up in seconds, ready for you to add your business objects. From Visual Studio: <em>File</em> -&gt; <em>New</em> -&gt; <em>Project</em>, select <em>Genesys Source Quick-Start</em>, and your stack framework is runnable and ready to code.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Cross-platform by default:</strong> Enable your business-objects to run truly everywhere: In web apps - In any mobile app - In desktop native apps - As middle-tier domain service - As Public API web services - even in client-side TypeScript.</td>
</tr>
</tbody>
</table>
Full-stack by default: Other frameworks focus only on the UI, only in the web, only for data access. Genesys Framework includes Database projects (SSDT), Middle Tier projects (.NET Standard) and projects for all major apps (MVC, Web API, UWP, WPF).

How do I get it?

The Genesys Source Framework is available where you need it most:

From Microsoft:
- In Visual Studio on Tools -> Extensions and Updates

From GitHub:
- On GitHub at http://github.com/genesyssource

From Genesys Source:
- At Genesys Source at http://cloud.Genesyssource.com

1: Genesys Framework: One Framework – Your Data – Any Platform

Pre-requisites

To get the most out of the Genesys Framework, 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/

Installing the Genesys Framework

The Genesys Framework is free open-source on GitHub and available as a Zip download, a Vsix download or through Visual Studio Marketplace…installable with any method that you prefer most.

The easiest way to get the Genesys Framework is to download our Quick-Start projects for MVC, Web API, WPF and Universal. The Quick-Start projects are a small starter with the basics to get you running fast.

Get as Zip, Vsix or directly in Visual Studio

1. Get

   Download Zip
   GenesysSource.com

   Download Vsix
   VisualStudioMarketplace.com

   Install in Visual Studio
   Tools -> Extensions and Updates

Framework for MVC

   Download Framework-for-MVC.sln
   Download Framework.MVC.vsix

   Tools -> Extensions and Updates -> Search: GENESYS -> Download
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework for WPF</td>
<td>Download Framework-for-WPF.sln</td>
<td>Download Framework.WPF.vsix</td>
<td>Tools -&gt; Extensions and Updates -&gt; Search: GENESYS -&gt; Download</td>
</tr>
<tr>
<td>Framework for Core</td>
<td>Download Framework-for-Core.sln</td>
<td>Download Framework.Core.vsix</td>
<td>Tools -&gt; Extensions and Updates -&gt; Search: GENESYS -&gt; Download</td>
</tr>
</tbody>
</table>

### 2 Install

- **Extract Zip**
  - If downloaded as Zip: Extract to a local folder
- **Run Vsix**
  - If downloaded as a VSIX, double-click the .vsix file and follow instructions
- **Close Visual Studio to Install**
  - Close/Reopen Visual Studio to allow Extensions and Updates to install the Genesys Quick-Start

### 3 Open

- **Open Solution File (.sln)**
  - If downloaded as Zip: Extract and Open .sln file
- **Create Solution**
  - File -> New Project -> C# -> Genesys Quick-Start

---

Copyright © Genesys Source
Uninstalling the *Genesys Framework*

If the *Genesys Framework* was downloaded via Zip file, no uninstallation is necessary.

For installs through the Visual Studio Marketplace or in Visual Studio, follow these steps to uninstall the Genesys Framework from your IDE:

1. **Open Visual Studio**
2. **Extensions and Updates**
3. **Click Uninstall and Close Visual Studio**
2: Uninstalling the Genesys Framework

What is in the Genesys Framework?
The Genesys Framework includes everything you need to build your business object framework quickly and with a minimal learning curve.

A Genesys Framework app includes full-stack projects for your application. From the database (SSDT), to data objects (EF), to models (.NET Core or Framework), exposed in any .NET application type such as MVC or UWP.

3: Genesys Framework Code and Runtime

### Visual Studio Project

<table>
<thead>
<tr>
<th>Framework.WebApp</th>
<th>MVC Web App with all CRUD and Search operations for a Customer entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Points of interest are:</td>
</tr>
<tr>
<td></td>
<td>✔️ \App\Data\ConnectionStrings.json – Database connection information</td>
</tr>
<tr>
<td></td>
<td>✔️ \Views\Home\index.cshtml – Home Page</td>
</tr>
<tr>
<td></td>
<td>✔️ \Controllers\Customer\CustomerSearchController.cs – Processes all customer search requests</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Framework.WebServices</th>
<th>Web API web services with all CRUD and Search operations for a Customer entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Points of interest are:</td>
</tr>
<tr>
<td></td>
<td>✔️ \App\Data\ConnectionStrings.json – Database connection information</td>
</tr>
<tr>
<td></td>
<td>✔️ \Views\Home\index.cshtml – Home Page</td>
</tr>
<tr>
<td></td>
<td>✔️ \Controllers\Customer\CustomerSearchController.cs – Processes all customer search requests</td>
</tr>
</tbody>
</table>
**Framework.UniversalApp**  
UWP Cross-Platform App with all CRUD and Search operations for a Customer entity.  
Points of interest are:  
- App\Data\ConnectionStrings.json – Database connection information  
- MainPage.xaml – Home Page  
- Pages\Customer\CustomerSearch.xaml – Processes all customer search requests

**Framework.DesktopApp**  
WPF Desktop App with all CRUD and Search operations for a Customer entity.  
Points of interest are:  
- App\Data\ConnectionStrings.json – Database connection information  
- MainPage.xaml – Home Page  
- Pages\Customer\CustomerSearch.xaml – Processes all customer search requests

**Framework.Models**  
Cross-platform PCL containing bindable screen models for MVC, WPF, UWP, WebForms, WinForms, Xamarin.  
Points of interest are:  
- \Customer\CustomerModel.cs – View Model for Customer business object

**Framework.Interop**  
Cross-platform PCL containing interfaces, to ensure all tiers share the same signature.  
Points of interest are:  
- \Customer\ICustomer.cs – Interface ensuring compatibility between all Customer objects

**Framework.DataAccess**  
Entity Framework data access objects, providing CRUD operations for Customer.  
Points of interest are:  
- \Customer\CustomerInfo.cs – Data Access Object for Customer business object

**Framework.Database**  
SSDT database containing all T-SQL for tables, views, stored procs, schemas, users.  
Points of interest are:  
- \Tables\Customer\Customer.sql – Customer table  
- \Views\CustomerCode\CustomerInfo.sql – View that connects table and code  
- \Stored Procedures\CustomerCode\CustomerInsert.sql – Stored procedure that inserts to customer table

---

2: Genesys Framework .NET Projects

**Running the Framework.Test Integration Tests**

All products contain Framework.Test, an integration test project that tests your objects and support classes. To run all tests in the solution:

1. Open your solution, i.e. Framework-for-MVC.sln Visual Studio solution file
2. Right-click the solution and click Rebuild Solution
3. Open the Test Explorer window  
   - Test -> Windows -> Test Explorer  
   - Click Run All to run all tests  
   - All tests should execute successfully

---

Copyright © Genesys Source
4: Running Framework.Test

Debugging the Genesys Framework

The Genesys Framework contains App and Services projects that host your application. These Apps can be debugged using standard .NET debugging techniques in Visual Studio Community (or greater.)

Debugging Framework.WebApp (MVC) and Framework.WebServices (Web API)

To debug your Framework for MVC and Framework for Web API app, follow the procedures below:

1. Open the Solution I.e. Framework-for-MVC.sln

2. Set Breakpoint in CustomerSearchController.cs

3. Set as StartUp Project and Press F5 to Run

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

6. Navigate to and open Framework.WebApp\Controllers\CustomerController

7. Set a breakpoint in the `Search()` method
   
   ```csharp
   public ActionResult Search(CustomerModel data)
   {
       var model = new CustomerSearchModel();
       var searchResults = CustomerInfo.GetByAny(data).Take(25);
   }
   ```

8. Right-click Framework.WebApp or Framework.WebServices project -> click Set as StartUp Project

9. Press F5 or `F5` to run
   - WebApp Url: `http://localhost:30001/`
   - WebServices Url: `http://localhost:30002/`

10. Home/Index.cshtml should display

4. Enter First (x) and Last (o)
    -> Click Search

5. Step-in to CustomerInfo.cs

6. See Customer search data before returning to View

Hint for LocalDB: Check SSMS Server (LocalDb)\MSSQLLocalDB if FrameworkData_Primary.mdf is locked.
11. Search from the home header
12. Enter a single letter (x) into First Name and a single letter (o) into Last Name
13. Click Search to hit breakpoint

14. Press F10 to step-over to `if (searchResults.Any())`
15. Select `searchResults`
16. Press SHIFT+F9 to see the Quick Watch window

17. Continue execution by pressing F5 to see the search complete

5: Debugging Framework for MVC and Web API

**Debugging Framework.UniversalApp (UWP) and Framework.DesktopApp (WPF)**

To debug your Framework for Universal and Framework for WPF app, follow the procedures below:

1. Open the Solution
   i.e. `Framework-for-Uwp.sln`

2. Set Breakpoint in
   `CustomerSearch.xaml.cs`

3. Set a breakpoint in the `Process()` method
   ```csharp
   public override async Task<WorkerResult> Process()
   {
       ... BindModel(MyViewModel.MyModel);
   }
   ```

4. Right-click `Framework.UniversalApp` or `Framework.DesktopApp` project -> click Set as StartUp Project

5. Right-click solution -> Configuration Manager -> Check Build and Deploy

6. Press F5 or ➪ to run

7. `MainPage.xaml` should display

Enter First (x) and Last (o)

Step-in to `CustomerInfo.cs`

See Customer search data before returning to View
8. Click Search icon on left
9. Enter a single letter (x) into First Name and a single letter (o) into Last Name
10. Click Search to hit breakpoint

11. Press F10 to step-over to BindModel(MyViewModel.MyModel);
12. Select MyViewModel.MyModel
13. Press SHIFT+F9 to see the Quick Watch window
14. Continue execution by pressing F5 to see the search complete

6: Debugging Framework for UWP and WPF

The **Framework.Database project and FrameworkData database**

**Genesys Framework** includes the **Framework.Database project**, which contains a micro-database as a go-between your database and your new code stack projects. Don’t have an existing database? No problem, as the **Genesys Framework** default operates 100% on the **FrameworkData database** that **Framework.Database** creates.

**Framework.Database** is Loose-coupled to your SQL Tables using Views

The **Genesys Framework** pulls data using Entity Framework, which can be tight-coupled directly to SQL Tables, or loose-coupled to SQL Views and Stored Procedures.

Out of the box, **Genesys Framework**:

- Connects to the **FrameworkData database**
- Selects data from SQL Views, i.e. **FrameworkData.CustomerCode.CustomerInfo**
- Insert, update and delete through SQL Stored Procedures, i.e. **FrameworkData.CustomerCode.CustomerInsert**
- **Framework.DataAccess** project contains Repository and Data objects for the data, i.e. **Customer\CustomerInfo.cs**
- In the App project, data is exposed as View Model objects, i.e. **Customer\CustomerModel.cs**

"Your Site, Service or App with Customer Pages"
About Framework.Database (SSDT)

The Framework.Database is a SQL Server project built on SQL Server Data Tools (SSDT). This project is responsible for:

1. Holds T-SQL for tables, schemas, indexes, constraints, users and roles
2. Holds and runs the PreDeployment and PostDeployment scripts
3. DB Compare the Framework.Database project to the FrameworkData database
4. Publishes the Framework.Database project to the FrameworkData database

Once deployed, you test the FrameworkData database as any other SQL Server database. Select from the Customer tables and CustomerCode views. Insert, update and delete from the CustomerCode stored procedures.

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.
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. Open your Solution i.e. Framework-for-MVC.sln
2. Connect to your database in SQL Object Explorer
3. Extract your database schema to a .dacpac file
1. Open the Framework-for-MVC.sln Visual Studio solution file
   - Default: C:\Source\Framework-for-MVC.sln
2. Click View -> SQL Server Object Explorer
3. Enter connection info to your database
4. Click Connect to add the connection
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

5. Open View

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

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

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

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
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. 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\Customer\Customer.sql

3. Add a new field: NickName
   - [NickName] NVARCHAR (50) CONSTRAINT [DF_Customer_NickName] DEFAULT ('') NOT NULL

4. Open CustomerInfo.cs in

9: Pulling your data through the CustomerInfo object

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'

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
### Publishing the Genesys Framework to IIS and SQL Server

For the Genesys Framework to function in your dev or production environments, you minimally need:

1. **Framework.Database project published to a SQL Server or SQL Express**

---

<table>
<thead>
<tr>
<th><strong>Publish FrameworkData to SQL Server</strong></th>
<th><strong>Framework.DataAccess</strong></th>
<th><strong>Add NickName to CustomerInfo</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Open SSDT publish screen: Framework.Database\Publish\PublishToDev.publish.xml - Ensure Target database connection is correct</td>
<td>9. Open CustomerInfo.cs - Framework.DataAccess\Customer</td>
<td>10. Add NickName by Copy/Paste the following property: <code>public string NickName { get; set; } = string.Empty;</code></td>
</tr>
<tr>
<td>7. Click Generate Script and review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Click Publish to push changes to SQL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### 7 Add NickName to ICustomer & Models

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)

#### 8 Add NickName to Framework.WebApp Search

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

#### 9 Run!

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
2. At least one Presentation Tier project, such as Framework.WebApp, published to an IIS Server

**Publishing Framework.Database (SSDT) to a SQL Server**

This procedure describes publishing the `Framework.Database` SSDT project to your SQL Server. The `Framework.Database` project holds all database objects for the `FrameworkData` database. Including tables, schemas, logins, users, views, stored procedures, etc.

1. Open the framework solution you wish to publish, for example: `C:\Repos\Framework-for-Mvc.sln`
2. Build `Framework.Database` project to ensure no errors
3. Open the Dev publish file `\Publish\PublishToDev.publish.xml`
4. Click *Edit* button
5. Change *Server Name* field to be the name of your SQL Server (i.e. `Dev-Sql-16`)
6. Click *OK*
7. Click the *Save Profile* button to save your changes
8. Click *Generate Script* to see the change script (no database changes will be applied.)
9. Click *Publish* to apply changes to the `FrameworkData` database

Your `Framework.Database` project is now in sync with your SQL Server

10: Publishing Framework.Database (SSDT) to SQL Server
Publishing Framework.WebApp (MVC) to an IIS Web Server

This procedure outlines how to publish the Framework.WebApp ASP.NET MVC project from Visual Studio to the IIS Web Server.

1. **Build**
   - Open the MVC solution you wish to publish, for example: `C:\Source\Framework-for-Mvc.sln`
   - Build the solution to ensure no errors

2. **Set Target Path**
   - Right-click the project and click Publish
   - Click the Settings... link in the Publish window
   - Change Target Location to be the path to your web project (default is local drive), for example: `\Dev-Web-16\WebSites\Framework.WebApp`
   - Click Save

3. **Publish**
   - Click Publish to publish the project to your development web server

The MVC project has now been published to your IIS Web Server.
Publishing Framework.WebServices (Web API) to an IIS Web Server

This procedure outlines how to publish the Framework.WebServices ASP.NET Web API project from Visual Studio to the IIS Web Server.

1. Build
2. Set Target Path
3. Publish

1. Open the MVC solution you wish to publish, for example: C:\Repos\Framework-for-Mvc.sln
2. Build the solution to ensure no errors
3. Right-click the project and click Publish
4. Click the Settings... link in the Publish window
5. Change Target Location to be the path to your web project (default is local drive), for example: \Dev-Web-16\WebSites\Framework.WebServices
6. Click Save
7. Click Publish to publish the project to your development web server

The Web API project has now been published to your IIS Web Server.

Tech and Code Aspects of the Genesys Framework

The Genesys Framework is a .NET Framework and Core stack containing C#, EF, SSDT, T-SQL, MVC, Web API, WPF, UWP, JavaScript, CSS and HTML. This section aims to explain some key tech aspects, to enable you to run and alter the projects with minimal learning curve.

Database Connections in App_Data

All projects communicate to SQL Server and SQL Express through the "DefaultConnection" connection string located in the following two files:

\App_Data\ConnectionStrings.Debug.json

\App_Data\ConnectionStrings.Release.json

Both include identical entries, but configured for different environments. For example: Dev SQL Server might be called DatabaseServer.dev.GenesysSource.com while Production is called DatabaseServer.prod.GenesysSource.com. Here is an example of the DefaultConnection connection string:
"DefaultConnection": "data source=DatabaseServer.test.GenesysSource.com; initial catalog=FrameworkData; user id=TestUser; password=57595709-9E9C-47EA-ABBF-4F3BA1B0D37; Multipleactiveresultsets=True; Application Name=GenesysFramework;"

13: DefaultConnection connection string

**Web Service Connections in App_Data**

All native-client project, such as Framework.UniversalApp (UWP) and Framework.DesktopApp (WPF) communicate to their Web API back-ends through the "MyWebService" application setting located in the following two files:

\App_Data\AppSettings.Debug.json

\App_Data\AppSettings.Release.json

Both include identical entries, but configured for different environments. For example: Dev might be called sampler.dev.GenesysSource.com while Production may be called sampler.GenesysSource.com.

Here is an example of the MyWebService app setting:


14: MyWebService application setting

**Framework.DataAccess pulls data through Framework.Database SQL Views**

The Framework.DataAccess project employs EF Core for SQL Server data access. By default, Framework.Database is configured to:

- Select data through a (1) SQL View
- Insert, Update and Delete data through three (3) SQL Stored Procedures

*Why? Because your code is now Loosely-coupled to your data through SQL Views.*

**SQL Views for Selects**

<table>
<thead>
<tr>
<th>Framework.DataAccess C# EF Code</th>
<th>Framework.Database SQL View</th>
<th>FrameworkData SQL Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="C# EF Code" /></td>
<td><img src="image2" alt="SQL View" /></td>
<td><img src="image3" alt="SQL Table" /></td>
</tr>
</tbody>
</table>

15: SQL Views for Selects

**SQL Stored Procedures for Inserts, Updates and Deletes**

<table>
<thead>
<tr>
<th>Framework.DataAccess C# EF Code</th>
<th>Framework.Database SQL SP</th>
<th>FrameworkData SQL Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4" alt="C# EF Code" /></td>
<td><img src="image5" alt="SQL SP" /></td>
<td><img src="image6" alt="SQL Table" /></td>
</tr>
</tbody>
</table>
16: SQL Stored Procedures for Inserts, Updates and Deletes

Why the Genesys Framework?

The Genesys Framework was built out of frustration with the Copy-paste Anti-pattern in our daily software engineering lives. Boomerang bugs, bloated classes, inconsistent coding standards made development slow and tedious. Most software engineers know of good practices, some have even built reusable stacks…but inevitably the project would not be approved or completed.

We set out to make code reuse fast, easy and with a minimal learning curve.

Why build reusable code?

Code reuse is an important theme in many of today’s 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 code full-stack, cross-platform 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:

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.

**Getting Help**

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

<table>
<thead>
<tr>
<th>Contact Genesys Source</th>
<th>Help and Guidance</th>
<th>On Social</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:help@genesyssource.com">help@genesyssource.com</a></td>
<td>[Azure] Try Cloud Dev Environment Free</td>
<td></td>
</tr>
</tbody>
</table>