Cloud Web Environment
eBook
Getting Help

Pre-requisites ........................................................................................................................................ 5
How-to Get the Cloud Web Environment .............................................................................................. 5
Installing the Cloud Web Environment .................................................................................................. 6
Stopping (deallocate) Cloud Web Environment for near-zero costs ...................................................... 7
Uninstalling the Cloud Web Environment .............................................................................................. 8
What is in the Cloud Web Environment? .............................................................................................. 9
Running the Framework.Test Integration Tests ....................................................................................... 11
Debugging the Genesys Framework ...................................................................................................... 13
  Debugging Framework.WebApp (MVC) and Framework.WebServices (Web API) ............................... 13
  Debugging Framework.UniversalApp (UWP) and Framework.DesktopApp (WPF) ........................... 14
The Framework.Database project and FrameworkData database ......................................................... 15
  About Framework.Database (SSDT) .................................................................................................. 16
Re-wire Framework.Database SQL Views to connect to your SQL Tables ............................................ 16
Add a new Field/Property to the Customer object ............................................................................... 18
Publishing the Genesys Framework to IIS and SQL Server .................................................................. 20
  Publishing Framework.Database (SSDT) to a SQL Server .............................................................. 20
  Publishing Framework.WebApp (MVC) to an IIS Web Server ....................................................... 21
Tech and Code Aspects of the Genesys Framework ............................................................................. 22
  Database Connections in App_Data ................................................................................................. 22
  Web Service Connections in App_Data ........................................................................................... 23
  Framework.DataAccess pulls data through Framework.Database SQL Views .................................. 23
Why the Genesys Framework? .............................................................................................................. 24
  Why build reusable code? .............................................................................................................. 24
  Why code full-stack, cross-platform business objects? .................................................................. 24
Getting Help ........................................................................................................................................ 25
Cloud Web Environment is the Cloud in a Can

The *Cloud Web Environment* is a trio of Virtual Machine servers that takes you from on-premises to the cloud in minutes. Includes a ready-to-run IIS Web Server, SQL Server 2017, plus a Visual Studio development machine.

**Tell me a little more about the Cloud Web Environment**

<table>
<thead>
<tr>
<th>What is it?</th>
<th>The <em>Cloud Web Environment</em> is a ready-to-run set of three (3) Azure Virtual Machines, pre-setup with no setup time needed:</th>
</tr>
</thead>
</table>
  ➢ Already-running Framework for MVC site  
  ➢ Already-running Framework for Web API services |
  ➢ Already-running FrameworkData DB |
  ➢ Genesys Framework open stack: Framework for MVC, Web API, WPF and Universal/Xamarin  
  ➢ Ready to code and publish to your Web Server and SQL Server |

**Why do I care?**  

**Cloud infrastructure in minutes:** Your entire web infrastructure is up in minutes, ready for you to add your web sites, web services and databases. Provision the VMs, copy your web site, restore your database and your web site is in the cloud and publicly available.
On-prem to Cloud with no learning curve: The cloud brings too many options and flavors, when most want to start small by mirroring their on-premises servers on the cloud. The Cloud Web Environment is a standard Web Server + SQL Server setup, that matches most server rooms.

Upgrade Tech to the latest like .NET Core & UWP: No experimentation and prototyping necessary. The Cloud Web Environment includes the Genesys Framework, our full-stack, cross-platform, open-source business object framework that is simple and runs the latest. Add your business objects, point to your database and your code stack is now running .NET Core!

How do I get it?
The Cloud Web Environment is available on Microsoft Azure in the Azure Portal and the Azure Marketplace:

From Microsoft:

1: Cloud Web Environment – Your Cloud in a Can

Pre-requisites
To get the most out of the Cloud Web Environment, the following skills are recommended:

✓ Basic IT Windows Server knowledge
✓ Basic IT Virtual Machine knowledge

To get the most out of the pre-installed Genesys Framework on the Cloud Web Environment, 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

How-to Get the Cloud Web Environment
This article summarizes how to get the 3-Server Cloud Web Environment on azure. The 3-servers of the Cloud Web Environment are:

1. Ready-to-run IIS Web Server [Server 1 of 3]
2. Ready-to-run SQL Server 2016 [Server 2 of 3]
3. Ready-to-run Visual Studio [Server 3 of 3]

Important! Azure Infrastructure Costs apply any time a VM is running, even in a free month. Usage will be charged by Azure until the VM is in Stopped (deallocated) state. Turn on auto-shutdown if you wish to keep costs low.
1. Create your Microsoft Account for Free (if you do not have an account) - Go to: http://azure.microsoft.com - Click FREE ACCOUNT or Signup links - Follow instructions to create your account

2. Create your Azure Pay-as-you-Go Subscription for Free (if you do not have an subscription) - Go to: http://portal.azure.com - Click Subscriptions - Click Add -> Pay-As-You-Go - Follow instructions for a Free subscription

3. Go to http://portal.azure.com
4. Click the New + button
5. In the Search box, enter: GENESYS
Install/Purchase all 3 VMs below:
- Ready-to-run IIS Web Server [Server 1 of 3]
- Ready-to-run SQL Server 2016 [Server 2 of 3]
- Ready-to-run Visual Studio [Server 3 of 3]

1: Installing the 3 Cloud Web Environment Servers

Installing the Cloud Web Environment
This section details, more in depth, the process of installing a Virtual Machine onto Azure. If you are not familiar with Azure infrastructure setup, follow the directions below for a safe, low-cost setup.

1. Go to http://portal.azure.com
2. Click the New + button
3. In the Search box, enter: GENESYS
   For each of the 3 VMs, install by:
   1. Click the Cloud Web Environment Server VM you wish to install
   2. Click Create at the bottom.
4. Fill in the following Step 7 fields:
   - Name: Your VM Name (i.e. Dev-Sql-16)
   - VM disk type: HDD (normal) or SSD (solid-state)
   - User name: Admin user (i.e. Local Admin)
   - Password/Confirm: Password (i.e. 2827iRo2624x)
   - Subscription: Subscription to be billed monthly
   - Resource Group: Group to contain this VM
   - Location: Location of the datacenter
5. Select your desired VM Size in Step 2.
   Minimum system requirements are:
   - CPU: 1 core
   - Memory: 1.75gb
6. Click OK
7. If you are new to Azure VMs, leave Step 3 default. Azure will auto-setup your network.
8. To save costs, you can turn on Auto-shutdown, VM consumption is $0 when shutdown:
9. Click OK

10. Carefully review the summary page.
11. Click Purchase to install this Virtual Machine

12. Click Virtual Machines

Your new Cloud Web Environment server will show in the list when provisioning is complete.

2: Installing a Ready-to-Run Cloud Web Environment Server

**Stopping (deallocate) Cloud Web Environment for near-zero costs**

In the cloud, Infrastructure-as-a-Service (IaaS) products like Virtual Machines typically charge for consumption, and do not charge if the Virtual Machine is Stopped (deallocated.)

**Important!** When Stopped (deallocated), the VM OS disk is still charged a small amount for storage. To completely stop all charging, please refer to *How-to Uninstall the Cloud Web Environment*

Follow these steps to Stop/Dealocate your VMs, to ensure you are no longer charged for consumption:

1. portal.Azure.com
2. Virtual Machines
3. Stop
2. Click Resource Groups
3. Click the Resource Group that includes the VM

13. Click the desired Virtual Machine in the list
14. Click Stop to change state to Stopped (deallocated)

You will no longer be charged for the Virtual Machine.
You are still charged a small amount for disk storage

3: Stopping (deallocating) the Cloud Web Environment to reduce costs

**Uninstalling the Cloud Web Environment**

Uninstalling the Cloud Web Environment involves deleting all 3 Virtual Machines. The following items will be deleted as part of this procedure: 1. All 3 Virtual Machines and 2. All 3 OS Disks

**Important!** Backup your data, files, everything from any VM you wish to uninstall! The VMs will be deleted after this procedure.

1. **Select VM**
2. **Delete VM**
3. **Delete Disk**

**Critical!** Backup all data, files, documents before proceeding.
2. Click Resource Groups
3. Click the Resource Group that includes the VM
4. Click the desired Virtual Machine in the list
5. Click Delete to uninstall and remove the product from your cloud portal
6. Click Yes if you are sure you want to delete

You will no longer be charged for the Virtual Machine

15. Click the Resource Group to see the full list of resources
16. Click the Disk associated to the deleted VM
17. Click Delete to uninstall and remove the associated OS Disk.

You will no longer be charged for the Disk
What is in the **Cloud Web Environment**?
The *Cloud Web Environment* includes everything you need to go from on-prem to the cloud in minutes, with almost no learning curve.

Included in the Cloud Web Environment are three (3) Virtual Machines:

1. Ready-to-run IIS Web Server [Server 1 of 3]
2. Ready-to-run SQL Server 2016 [Server 2 of 3]
3. Ready-to-run Visual Studio [Server 3 of 3]
5: Cloud Web Environment and Genesys Framework

In addition to the three (3) servers of the Cloud Web Environment, our open-source, full-stack, cross-platform Genesys Framework is also pre-setup and ready-to-run. The Visual Studio projects included in the Genesys Framework are described below.

### Genesys Framework Visual Studio Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework.WebApp</td>
<td>MVC Web App with all CRUD and Search operations for a Customer entity.</td>
</tr>
<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>
<tr>
<td>Framework.WebServices</td>
<td>Web API web services with all CRUD and Search operations for a Customer entity.</td>
</tr>
<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>
<tr>
<td>Framework.UniversalApp</td>
<td>UWP Cross-Platform App with all CRUD and Search operations for a Customer entity.</td>
</tr>
<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>➢ \MainPage.xaml – Home Page</td>
</tr>
<tr>
<td></td>
<td>➢ \Pages\Customer\CustomerSearch.xaml – Processes all customer search requests</td>
</tr>
<tr>
<td>Framework.DesktopApp</td>
<td>WPF Desktop App with all CRUD and Search operations for a Customer entity.</td>
</tr>
<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>➢ \MainPage.xaml – Home Page</td>
</tr>
<tr>
<td></td>
<td>➢ \Pages\Customer\CustomerSearch.xaml – Processes all customer search requests</td>
</tr>
<tr>
<td></td>
<td>Points of interest are:</td>
</tr>
<tr>
<td></td>
<td>➢ \Customer\CustomerModel.cs – View Model for Customer business object</td>
</tr>
<tr>
<td>Framework.Interop</td>
<td>Cross-platform PCL containing interfaces, to ensure all tiers share the same signature.</td>
</tr>
<tr>
<td></td>
<td>Points of interest are:</td>
</tr>
<tr>
<td></td>
<td>➢ \Customer\Customer.cs – Interface ensuring compatibility between all Customer objects</td>
</tr>
<tr>
<td>Framework.DataAccess</td>
<td>Entity Framework data access objects, providing CRUD operations for Customer.</td>
</tr>
<tr>
<td></td>
<td>Points of interest are:</td>
</tr>
<tr>
<td></td>
<td>➢ \Customer\CustomerInfo.cs – Data Access Object for Customer business object</td>
</tr>
<tr>
<td>Framework.Database</td>
<td>SSDT database containing all T-SQL for tables, views, stored procs, schemas, users.</td>
</tr>
<tr>
<td></td>
<td>Points of interest are:</td>
</tr>
<tr>
<td></td>
<td>➢ \Tables\Customer\Customer.sql – Customer table</td>
</tr>
<tr>
<td></td>
<td>➢ \Views\CustomerCode\CustomerInfo.sql – View that connects table and code</td>
</tr>
<tr>
<td></td>
<td>➢ \Stored Procedures\CustomerCode\CustomerInsert.sql – Stored procedure that inserts to customer table</td>
</tr>
</tbody>
</table>

2: Genesys Framework Visual Studio .NET Projects
One Framework – Your Data – Any Platform

Pre-installed on the Cloud Web Environment is the Genesys Framework, 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.

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.
## Visual Studio Project

<table>
<thead>
<tr>
<th>Framework</th>
<th>Description</th>
<th>Points of interest</th>
</tr>
</thead>
</table>
| Framework.WebApp   | MVC Web App with all CRUD and Search operations for a Customer entity.      | ➢ \App_Data\ConnectionStrings.json – Database connection information
|                    |                                                                             | ➢ \Views\Home\Index.cshtml – Home Page                                                                      |
|                    |                                                                             | ➢ Controllers\Customer\CustomerSearchController.cs – Processes all customer search requests                  |
| Framework.WebServices | Web API web services with all CRUD and Search operations for a Customer entity. | ➢ \App_Data\ConnectionStrings.json – Database connection information
|                    |                                                                             | ➢ \Views\Home\Index.cshtml – Home Page                                                                      |
|                    |                                                                             | ➢ Controllers\Customer\CustomerSearchController.cs – Processes all customer search requests                  |
| Framework.UniversalApp | UWP Cross-Platform App with all CRUD and Search operations for a Customer entity. | ➢ \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. | ➢ \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. | ➢ \Customer\CustomerModel.cs – View Model for Customer business object                                        |
| Framework.Interop  | Cross-platform PCL containing interfaces, to ensure all tiers share the same signature. | ➢ \Customer\Customer.cs – Interface ensuring compatibility between all Customer objects                      |
| Framework.DataAccess | Entity Framework data access objects, providing CRUD operations for Customer. | ➢ \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. | ➢ \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     |

### 3: 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*
2. Rebuild All on the Solution
3. Click Run All in the Test Explorer Window
7. Open your solution, i.e. Framework-for-MVC.sln Visual Studio solution file
6. Right-click the solution and click Rebuild Solution
7. Open the Test Explorer window
   - Test -> Windows -> Test Explorer
8. Click Run All to run all tests
   - All tests should execute successfully
   Hint for LocalDB: Check SSMS Server (LocalDb)\MSSQLLocalDB if FrameworkData_Primary.mdf is locked.

7: 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:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 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 |   |

8. Open the *Framework-for-MVC.sln*
   Visual Studio solution file
   - Default: C:\Source\Framework-for-MVC.sln
9. Navigate to and open
   Framework.WebApp\Controllers\CustomerController
10. Right-click Framework.WebApp or Framework.WebServices project -> click Set as StartUp Project
11. Press F5 or Ctrl+F5 to run
12. Home/Index.cshtml should display
8: 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 as StartUp Project and Press F5 to Run
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) - > Click Search

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

9: 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
10: Data passing through Framework objects

**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.

11: Selecting from the Customer table

**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

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...

7. 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

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
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 Framework-for-MVC.sln
2. Open Table & Add Column Tables\Customer\Customer.sql
3. Open View & Add Column Views\CustomerCode\CustomerInfo.sql
4. Navigate to and open Customer view:

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

7. Alias any missing fields with Default Values

8. Publish FrameworkData to SQL Server

9. Run Framework.WebApp to pull your customer data
- Default: C:\Source\Framework-for-MVC.sln

3. Add a new field: NickName

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

5. Add the NickName field:

   ```
   C.[NickName]
   ```

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

7. Add NickName to ICustomer & Models

8. Add NickName to Framework.WebApp Search

9. Run!

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

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

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

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
13. Add NickName to all dependent models (CustomerModel and CustomerSearchModel)

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
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
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
2. Set Target Path
3. Publish

1. Open the MVC solution you wish to publish, for example: C:\Source\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.WebApp
6. Click Save
7. 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. 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.

15: Publishing Framework.WebServices (Web API) to 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-4F3BAA1B0D37; Multipleactiveresultsets=True; Application Name=GenesysFramework;"
16: 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:


17: 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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

18: 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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
19: 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>

**GENESYS SOURCE**

22431 Antonio, Suite B160-843
Rancho Santa Margarita, CA 92688
+1 949.544.1900
www.genesyssource.com
help@genesyssource.com