Chapter 1

Haystack Overview

Table of Contents

Haystack Overview

Haystack Code Generator for .NET is a utility that will help you build N-Tier, Service-Oriented Data Access Classes to perform select, search, add, edit, and delete functionality. A Data Access Class (DAC) is a wrapper around a single table, view, or stored procedure in a database. This wrapper class contains all of the CRUD (Create/Read/Update/Delete) logic for a table. For a database view the DAC is a read-only class you use to call the view. For a stored procedure this DAC contains all of the input and output parameters needed to communicate with the stored procedure.

Haystack includes tools, templates, samples, and documentation that simplify and accelerate the delivery of Angular, jQuery, WPF, MVC, and Web Form applications for C# developers. Haystack will build n-tier classes for you in a fraction of the time of other products. By utilizing Haystack tool you can standardize your company's development efforts, thereby increasing productivity and maintainability.

Haystack will provide a significant productivity boost to your development team in developing data access routines and in maintaining them. Most developers do not have the time or the expertise to use all of the latest component development techniques available today. By using Haystack you are taking a component architecture that is already designed and applying it to your application. This will significantly increase your reliability, speed of development and will reduce the costs of producing the application.

This chapter gives you an overview of the philosophy and the goals of Haystack.

Philosophy

To get the most out of the PDSA method of application development you should understand some key concepts:

Coding standards are vital to the success of a project. All programmers must follow the standards set forth in these documents. If you adhere to these standards, your code will be more consistent and all programmers will be able to read all other programmers code. **NOTE**: The PDSA standards documents are contained in the \Documentation\Standards folder in a Microsoft Word format in the installation folder of this product.

When developing applications, the pages, forms or user controls should NOT know anything about table structures, data access methods, or where the data is coming from. Instead all the logic to retrieve, add, edit and/or delete from any table, view and/or stored procedure in your database should be encapsulated into its own Data Access Class (DAC). This makes sure that you are not duplicating code throughout your application. Once you create a class/method for accessing a specific object in your database, you ensure that class/method is used consistently throughout your application.

All **Data Classes** are further removed from the pages, forms or user controls through the use of a **View Model** class. This View Model class is directly used by the page, window or user control.

Several classes are generated from each table, view or stored procedure. All these classes are contained in one .generated.cs file and marked as "partial". A separate .cs file is generated for each class, and is also marked as "partial". This is where you can add your own code to extend the generated classes. You will learn more about the classes generated in a later section of this chapter.

The use of partial classes for each of the generated classes allows you to regenerate the underlying DACs at any time via the Haystack while preserving your customizations.

Using Haystack Generated Code

Haystack generates generic .NET code that can be used in any version of .NET from 4.52 and later. The generated code relies on a few things in order to work.

1. Some PDSA .dlls need to be added to your Visual Studio project.
2. A reference to System.Linq.Dynamic.dll (supplied with Haystack) needs to be added to your Visual Studio project.
3. Configuration entries need to be added to your Web.Config or App.Config file in your Visual Studio project.
4. To distribute your .NET application you will need to create a runtime license from Haystack. A runtime license can only be generated from a purchased copy of Haystack, not the trial version.

Goals of Haystack

When PDSA, Inc. set out to build a data access class code generator we had several goals in mind. Obviously ease-of-use and rapid application development were topmost in our minds. But there were some others that also came about:

- Supply templates that the code generator uses in combination with information from a repository (database) to create complete data access classes and generate CRUD stored procedures.
- Allow the programmer to modify these templates.
- Be able to quickly generate all of the data access methods for a particular object in a database.
- Be able to generate 100% of the data access code an application can use to SELECT, SEARCH, INSERT, UPDATE and DELETE data.
- Generate standard business rules inferred from the schema of a table.
- Supply several reusable classes and DLLs that can be of immediate use in your programs.
- Supply examples of how to use the code generated from Haystack.

What Haystack Can Create

Haystack helps you generate code that would otherwise be very time-consuming and boring to write. The following items can be generated by the Haystack.

Business/Manager Class

A **Business/Manager Class** performs the (CRUD) retrieval and modification of data in a table in a database. This retrieval and modification can be performed by submitting SQL statements, or by submitting the name of a stored procedure (complete with parameters) from within this class.

Entity Class

An **Entity Class** is a list of properties that match to the fields of your table or view, or to the parameters of a stored procedure. This entity class is designed to be transferred across a service boundary through a Web API or WCF service. Data annotations can be applied to this class to enforce business rules.

Search Class

A **Search Class** is a list of properties used to facilitate searching from a page, form or user control. This class typically has properties that closely match your Entity Class, yet no business rules are applied to the properties of this class.

Validation Class

A **Validation Class** contains validation rules inferred from your table schema. You may also add additional business rules. Like with Business/Manager Classes you are able to regenerate these validation classes if your schema changes while preserving any custom code you have written.

View Model Generation

PDSA has standardized on the Model-View-View-Model (MVVM) approach for all of our application development. A view model class will be generated that all page, forms or user controls (the View in MVVM) will communicate with. The view model class communicates with the Business/Manager Classes and the Entity Classes (the Model in MVVM).

Stored Procedures

Haystack can generate stored procedures that read and modify data in a database table. These stored procedures are then called from the generated Data Access Classes.

WPF CRUD User Control

For each table in your schema you may generate a standard add/edit/delete user control for WPF. This user control will give you a good starting point for developing your application’s screens.

Web Forms CRUD Page

For each table in your schema you may generate a standard add/edit/delete page for Web Forms. This page will give you a good starting point for developing your application’s screens.

MVC CRUD Pages

For each table in your schema you may generate a standard add/edit/delete pages for ASP.NET MVC. These pages will give you a good starting point for developing your application's screens.

Web API Services Generation

Haystack is designed with a Service Oriented Architecture (SOA) in mind. Our Entity Classes, Business/Manager Classes and Validation Classes all know how to work with Web API service applications. We generate Get, Put, Post and Delete methods in a Web API.

WCF Services Generation

Haystack is designed with a Service Oriented Architecture (SOA) in mind. Our Entity Classes, Business/Manager Classes and Validation Classes all know how to work with WCF service applications. We generate Interfaces, Services and Response classes that can be used in MVC, Web Form, WPF, Windows Forms, HTML 5, jQuery, Angular Windows Phone or Windows Universal applications.

Why Use Data Access Classes

There are many reasons to use data access classes in your database programming.

- Eliminates SQL from your pages, forms, controllers, and user controllers
- Allows entity classes to be used across a service like a Web API or WCF service.
- Hides database implementation
- Hides data access method from the front-end
- Less changes to front end code when column is changed
- IntelliSense listing of properties/column names
- Faster development, less bugs
- Reusable components that can be used from C#, Visual Basic, or any other .NET language.

- Code generation leads to more bullet proof and tested code. This leads to less bugs and faster development.

A Haystack Project

Haystack uses a **Project** as a pointer to a single database/catalog within a server. Each project has certain attributes about it. For example, a project will be able to generate classes for tables, views and stored procedures. Business/Manager Classes can use either Dynamic SQL or Stored Procedures. Any project can point to a set of templates to be used to generate code. These are just a few of the attributes that you will setup to help you generate your data access classes.

Summary

Haystack helps your company develop your .NET applications much faster than coding by hand. In addition to data classes you can generate WPF windows and user controls, and Web Form and MVC pages.

Chapter Index