

## Code Generation and Frameworks

By Andres Aguiar, Chief Software Architect for DeKlarit

Code generation is a powerful concept that is acquiring big momentum in the .NET world. The code created by code generators is designed to run on top of different types of frameworks, and it may be used as a framework itself.

Frameworks usually create an abstraction layer on top of a previously existing framework, simplifying its use. Once a domain is fully understood and the main use patterns are identified, it's possible to design a framework that lets you use most of the underlying one, greatly reducing the knowledge needed to use it. For example Windows Forms is a framework that abstracts the Windows API, which is another framework, and makes creating Windows UI applications much easier.

However, building a framework is not an easy task. Being complex artifacts, they are difficult to design, develop, document, maintain, and use. They usually require good documentation, training events, books, and community support.

When we designed DeKlarit we could have designed a new Data Access API that simplified the way you accessed data in .NET. Instead, we decided to follow the DataSet/DataAdapter approach, because, for the above reasons, we didn't want to create a new Data Access framework. We built an enhanced DataSet and DataAdapter, and provided a better experience around them, but we didn't build a new framework. Thus, DeKlarit users just need to know how to use plain ADO.NET DataSets in order to use the generated code.

When we began generating the presentation layer, we could have designed our own set of Windows Forms and ASP.NET controls. For the same reasons, we decided not to do that, but we instead partnered with Infragistics to use their NetAdvantage suite.

DeKlarit 3.5 will support authentication, authorization, caching and encryption using Microsoft's Enterprise Library. Once again, we could have built our own framework to do this.

The code that DeKlarit generates integrates all these frameworks, making it easy to develop sophisticated, visually appealing, manageable, secure, and scalable applications, without requiring the DeKlarit user to learn a DeKlarit-specific API.

In the meantime, instead of focusing on building our own framework, the DeKlarit team can focus on what it does best: letting you specify your application declaratively, and giving you enough options to use the generated components in the way that best suits your needs.