

codesis - ein Schablonen basierter Code-Generator

Ausgangssituation

Bereits in den siebziger Jahren wurde ein Phänomen in der Softwareentwicklung identifiziert und als „Softwarekrise“ beschrieben, bei dem sich gezeigt hatte, dass häufig die tatsächlich angefallenen Kosten / Aufwände stark von den vor Projektbeginn geschätzten Kosten abwichen. Zur Lösung dieses Dilemmas entstanden hoch entwickelte Programmiersprachen und ausgefeilte Entwicklungswerkzeuge mit grafischen Benutzeroberflächen (IDE). Aus dieser Zeit heraus entwickelte sich auch der erste Einsatz von (Code-) Generatoren zur Verarbeitung von häufig wiederkehrenden Anforderungen.

Allgemeines über Generatoren

Code-Generatoren sind im Allgemeinen Anwendungen, die automatisiert (Quell-) Code erzeugen. Gleichartige und immer wiederkehrende Codezeilen müssen mit dem Einsatz von Generatoren nicht mehr von Hand vorgenommen werden. Auch Routinearbeiten oder das Einfügen von fertigen Bausteinen via „Copy & Paste“ bzw. manueller Anpassung können reduziert werden.

Generatoren unterstützen bei gleichbleibenden und häufig wiederkehrenden Sachverhalten, wie zum Beispiel die Entwicklung einer Datenbankzugriffsschicht inkl. Umsetzung des Datenmodells in ein Datenbankschema und die Erstellung geeigneter Datenbankprozeduren und helfen bei der Durchsetzung unternehmensweiter, vereinheitlichter Programmierparadigmen.

Des weiteren sind Codegeneratoren häufiger Bestandteil von Modellierungswerkzeugen, die aus abstrakten Modellbeschreibungen Quellcode-Skelette in der gewählten Programmiersprache erzeugen können.

Ein Generator ist eine Software die Anwendungsprogramme aus Problembeschreibungen erzeugt, im Gegensatz zur traditionellen Programmierung. Per Definition ist ein echter Codegenerator ein Programm, das auf Basis der Entwickler-Spezifikationen vollständige und lauffähige Programme erzeugt.

Bei der Klassifizierung von Generatoren muss zwischen der Generierung von statischem und von dynamischen Code unterschieden werden. Bei der Generierung von statischem Code ist der Generierungsprozess zur Laufzeit bereits abgeschlossen. Beispiele hierfür sind unter anderem:

- Compiler, die aus Eingaben in Hochsprachen Programmcode in Maschinsprache erzeugen
- Konverter, die Code von einem Format in ein anderes umwandeln
- Dokumentationswerkzeuge wie ndoc oder Javadoc, die aus Quellcodes Dokumentationen im HTML-Format erzeugen
- Textsatzsysteme, wie TeX / LaTeX
- Modellübersetzer, die Code-Dateien auf der Basis von abstrakten Modellbeschreibungen heraus erzeugen (s. UML Tools)

Die Generierung von dynamischem Code kommt immer dann zum Einsatz, wenn der zu erzeugende Code von Parametern abhängt, die erst zur Laufzeit bekannt sind.

- Applikationsserver wie Apache Tomcat oder Microsoft IIS, die aus JSP / ASP Dateien (Eingebetteter Code in HTML Anweisungen) Programmcode erzeugen.
- Berichtsgeneratoren wie Crystal Reports oder Jasper Reports, die auf der Basis von Datensätzen aktuelle Berichte im HTML-, PDF-Format usw. erstellen.

codesis Überblick

codesis wird als schablonenbasierter Generator bezeichnet. Das heißt die zu generierende Ausgabe wird über eine (oder mehrere) Schablonen erzeugt. Diese Schablonen entsprechen in wesentlichen Teilen der Ausgabe, werden aber als Schablonen oder Templates bezeichnet, weil in Ihnen Platzhalter oder Steuerungsanweisungen enthalten sein können, die erst durch den Generator ausgefüllt bzw. ausgeführt werden.

Die Schablonen bestehen in der Regel aus zwei ineinander verwobenen Bestandteilen. Der eine Bestandteil ist der Code, der als Ausgabe eines Generatorlaufs erzeugt werden soll. Den anderen Bestandteil bilden die Steuerungsinformationen für den zu erzeugenden Code, sowie die Elemente, die die Platzhalter innerhalb des Codes ersetzen sollen.

Damit lässt sich der Inhalt der Schablonen wie folgt in zwei Bereiche einteilen:

- der Steuerungscode oder auch Makrocode genannt
- der eigentliche Quellcode, der als Zielcode bezeichnet wird

Schablonen können als mit Makroaufrufen angereicherter Quellcode definiert werden. Dabei werden Makrocode und Zielcode durch einfache Trennzeichen (Start und Ende-Tags) voneinander separiert. Der Makrocode kann dabei als fest in dem Sinne angesehen werden, dass hier nur bestimmte fest definierte und vorgegebene Programmiersprachen (die Makrosprache) zum Einsatz kommen. Bisher sind C#, Java und VB.NET zur Verfügung.

Der Zielcode hingegen ist vollständig variabel. Hier können beispielsweise einfache Texte oder auch Skripte für beliebige Programmiersprachen enthalten sein. Der Zielcode ist aus Sicht des Generators ein einfacher Textbaustein, dessen Syntax und Semantik für den Generator und seine Komponenten keine Rolle spielt. Die Schablone wird durch den Generator in generierten Sourcecode umgewandelt. Dieser generierte Sourcecode liegt in der gleichen Programmiersprache wie der Makrocode vor, der den Zielcode nur noch als Text, mit entsprechenden Ausgabeanweisungen versehen, enthält.

Damit der Generator den Makrocode ausführen bzw. die Platzhalter ausfüllen kann müssen dem Generator so genannte Metadaten übergeben werden. Die Struktur der Metadaten, die den Schablonen als Eingabe dient, kann frei definiert werden, lediglich eine Beschreibung in Form eines XML Schema Document (XSD) muss vorliegen. Die so definierte Struktur der Metadaten wird kurz als Strukturdaten bezeichnet. Über diese Strukturdaten kann der Aufbau der Metadaten sehr flexibel gestaltet und definiert werden. So können einfache Datenbeschreibungen mit nur wenigen Attributen bis hin zu kompletten UML Modellen in Form eines XML Exports als Metadaten definiert werden. Einschränkende Bedingung für die Strukturdaten ist, dass ihr Aufbau eine Umwandlung in die Makrosprache erlaubt.

Die Metadaten werden als Instanzen dieser Strukturdaten den Schablonen zur Verfügung gestellt und müssen alle relevanten Informationen für den Makrocode enthalten. Diese Instanzen sind XML Daten, die gegen das vorgegebene Schema (XSD) der Strukturdaten überprüfbar sein müssen, und werden als Projektdaten bezeichnet. Der Zugriff innerhalb des Makrocodes der Schablonen kann über die Umwandlung der Strukturdaten in Strukturen der Makrosprache direkt erfolgen.

Der Codegenerator ist darauf ausgerichtet, als Eingabe (Projektdatei) mit einer XML-Datei zu arbeiten. Das zur XML-Datei gehörige Schema (XSD) muss vorab definiert werden. Die Schema Informationen werden in XML serialisierbare Klassen der Makrosprache umgewandelt.

Mittels der Projektdatei wird durch den Generator aus den Schablonen die eigentliche Ausgabe, der TemplateOutput erzeugt.

Durch den Zwischenschritt der Umwandlung in Makrocode ist es möglich Strukturdaten und Schablonen in einer zusammenhängende Binärdatei abzulegen und als eigenständiges, binäres Auslieferungsobjekt zu betrachten. Ausgestattet mit einem speziellen Editor für die Projektdatei können diese ein eigenständiges Produkt darstellen.

codesis Schablonen

Aktuell bietet codesis ca. 70 bis 80 Schablonen für den .NET 3.0 und VS2005 (zur Unterstützung für .NET 2.0 liegen ca. 60 bis 70 Schablonen bereit). Dabei unterstützen die Schablonen die folgenden Bereiche:

Microsoft SQL Server

Erzeugen der Skripte zum Anlegen und Löschen der Tabellen und zugehörigen Schattentabellen inklusive Foreign Key Constraints. Die Schattentabellen protokollieren dabei sämtliche Veränderungen an den eigentlichen Tabellen, sie halten somit die technische Historie der Tabellen nach. Erzeugen der Skripte zum Anlegen und Löschen diverser Stored Procedures zum Datenzugriff. Bereitstellung der Skripte für Trigger zur Befüllung der Schattentabellen.

Microsoft Enterprise Library

Erzeugung der Datenzugriffsschicht auf Basis der Microsoft Enterprise Library. Für die Datenzugriffsschicht werden so genannte TableDataGateways erzeugt, die sämtliche Zugriff auf die Stored Procedures kapseln. Die TableDataGateways werden als partielle Klassen erzeugt, die somit leicht um individuelle Funktionalitäten erweitert werden können.

Windows Communication Foundation

Als Kommunikationsschicht für Applikationen-Server existieren Schablonen zur Unterstützung der Enterprise Services und Web-Services auf Basis .NET 2.0. Für das .NET Framework 3.0 wird eine Anwendungsarchitektur auf Basis der Windows Communication Foundation (kurz WCF) unterstützt in den

Kommunikationsausprägungen TCP / HTTPS und MSQM. Sämtliche Services sind damit über die Möglichkeiten der WCF sowohl synchron wie auch asynchron ansprechbar.

Visual Studio

Zur einfachen Compilierung werden die generierten Source-Files in VS Projekt- und Solution-Files gruppiert. Diese werden selbstverständlich selbst auch wieder automatisch erzeugt.

MSBuild

Zur Unterstützung automatischer Builds werden sowohl Skripte für MS Build und NAnt generiert.

Tools

codesis liegt in derzeit in zwei verschiedenen Varianten vor, einerseits die Commandline-Version und andererseits als codesis Studio. Das codesis Studio, ausgestattet mit einer graphischen Benutzerschnittstelle, ermöglicht die Verwaltung der Metadaten, Projektdatei und Templates. Beide Versionen liegen derzeit in Beta-Versionen vor und können bei Interesse angefragt werden.

Verfügbarkeit

Wir rechnen derzeit mit einer ersten Version für das erste Halbjahr 2008.

Essen 15.09.2007

idesis GmbH

Rellinghauser Straße 334F
D-45136 Essen
www.idesis.de