

Effizienter Entwickeln mit Templates

Rainer Stropek
software architects gmbh
MVP für Windows Azure

Abstract

Jeder Entwickler kennt Routineaufgaben, die notwendig aber lästig sind. Ein Projekt mit Referenzen ist anzulegen, das Grundgerüst für eine zusätzliche Stammdatenmaske muss erstellt werden, in der Datenzugriffsschicht müssen SQL-Statements zusammengebaut werden etc. Visual Studio und .NET bieten eine Reihe von Hilfsmitteln, mit denen man vom fleißigen Handwerker zur "Softwarefabrik" wird.

- In Minuten mit Templates eine eigene Bibliothek an Projekt/Dateimustern aufbauen und nahtlos in VS einbetten
- Vorteile und Möglichkeiten von Snippets
- T4-Templates, die "Magie", die bei Entity Framework den C#-Code erzeugt, in eigenen Projekten zur Laufzeit einsetzen
- Alle Konzepte werden an einem durchgängigen Beispiel demonstriert, das den Teilnehmern zur Verfügung gestellt wird



Vom Handwerk zum Produktionsprozess
Was ist eine Softwarefabrik?

Probleme "typischer" Softwareerstellungprozesse

- Verfügbarkeit von Softwareentwicklern/innen
- Kosten- und Zeitdruck
- Monolithische Systeme
- Einsatz zu allgemeiner Frameworks
Beispiel: .NET Framework vs. MS Access
- One-Off Development
Wiederverwendung bestenfalls auf Ebene einfacher Basisklassen oder Plattformkomponenten
- Geringer Reifegrad im Spezifikationsprozess
- Innovationszyklen bei Basistechnologien
- U.v.m.

Was ist eine Softwarefabrik?

A software factory is an organizational structure that specializes in producing computer software applications or software components [...] through an assembly process.

The software is created by assembling predefined components. Traditional coding, is left only for creating new components or services.

A composite application is the end result of manufacturing in a software factory.

Quelle: "Software Factory" in Wikipedia

Vorteile einer Softwarefabrik (1/2)

Kernkompetenzen der Mitarbeiter werden hervorgehoben

Fachlich orientierte Berater konfigurieren (Vokabular näher bei der jeweiligen Domäne des Kunden)
Softwareentwickler erstellen Basiskomponenten

Steigerung der Effizienz

Das Rad wird weniger oft neu erfunden

Steigerung der Qualität

Anspruchsvolle QS-Maßnahmen für Basiskomponenten

Vorteile einer Softwarefabrik (2/2)

Reduktion der Projektrisiken

Fachberater mit besserem Kundenverständnis
Höhere Qualität der Basiskomponenten

Steigerung des Firmenwerts

Systematisches Festhalten von Wissen über die Herstellung einer Familie von Softwarelösungen
Design Patterns, Frameworks, Modelle, DSLs, Tools

Vereinfachung des Vertriebs- und Spezifikationsprozesses

Konzentration auf projektspezifische Lösungsteile
Quick Wins durch Standardkomponenten

Was eine Softwarefabrik nicht will...

Reduktion des Entwicklungsprozesses auf standardisierte, mechanische Prozesse

Im Gegenteil, mechanische Prozesse sollen Tool überlassen werden

Verringerung der Bedeutung von Menschen im Entwicklungsprozess

„Handwerk“ der Softwareentwicklung wird nicht gering geschätzt sondern gezielt eingesetzt

Entwicklung von Frameworks statt Lösungen für Kunden

Software Factories → Economy of Scope

Economy of Scale

Multiple implementations (=Copies) of the same design

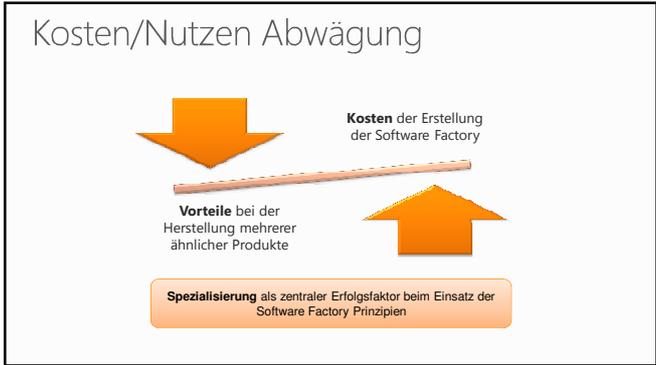
- Mehr oder weniger mechanische Vervielfältigung von Prototypen
- Massengüter
- Software (z.B. Auslieferung auf Datenträger)

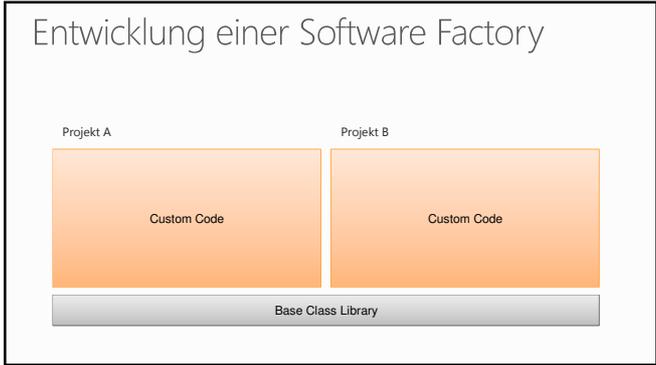
Economy of Scope

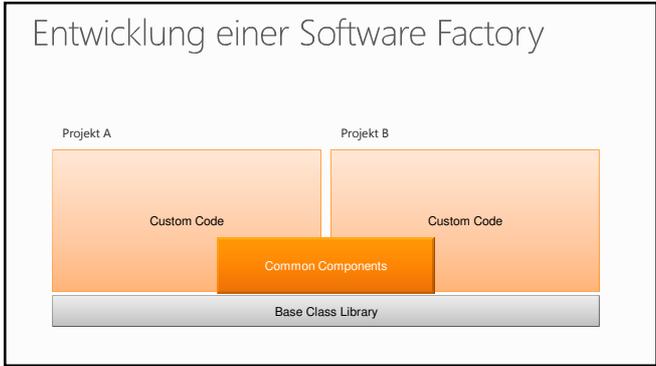
Production of multiple designs and their initial implementations

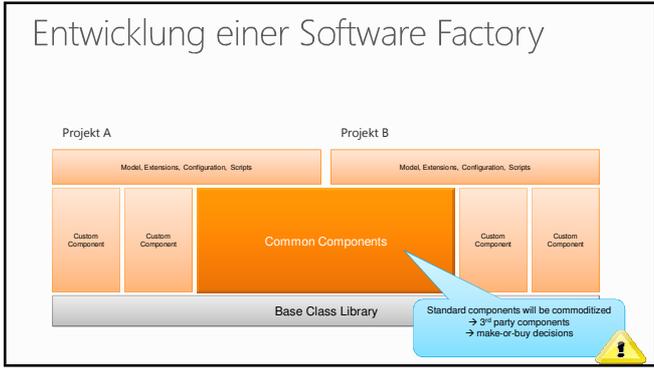
- Ähnliche Designs (=Familien von Anwendungen) auf Grundlage gemeinsamer Techniken und Technologien
- Individuelle physische Güter (z.B. Brücken, Hochhäuser)
- Individualsoftware, Softwareplattformen (vgl. PaaS)

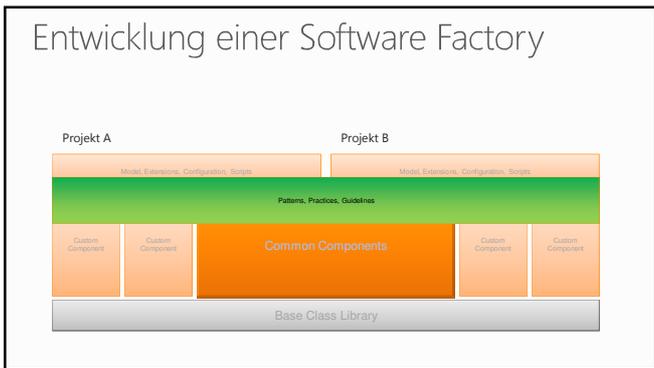
Quelle: Greenfield J. Short W. Software Factories











Werkzeuge

- Klassenbibliotheken
 - Dokumentation
 - Statische Codeanalyse
- Codegeneratoren
 - Vorlagen
 - Patterns in Form von Assistenten
- Domänenspezifische Sprachen
 - XML-basierend oder individuell (Compiler-Compiler)
 - Compiler (Codegenerator) vs. interpretiert
- Scriptsprachen
- Anwendungsmodularisierung
- Prozessautomatisierung
 - Qualitätssicherung
 - Build

Beispiele aus der Microsoft-Welt

- MS Framework Design Guidelines
 - Sandcastle
 - StyleCop, Code Analysis
- **Codegeneratoren**
 - T4, ANTLR StringTemplates
 - Visual Studio Templates
- Domänenspezifische Sprachen
 - XAML (UI und Workflows), EF
 - ANTLR (Compiler-Compiler)
- DLR, Project „Roslin“
- MEF
- Prozessautomatisierung
 - Visual Studio Unit Tests
 - TFS Buildautomatisierung

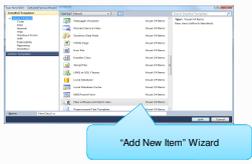


Einleitung in VS Templates

Vorlagen für
 Projekte
 Projektelemente

Bestehen aus
 Vorlagendateien
 Template Metadata Files (.vstemplate)
 (Optional) Individuelle Wizards
 (Custom Code; siehe `Microsoft.VisualStudio.TemplateWizard.IWizard`)

Manuelle Verteilung oder Installation
 VSIX Dateien



Individuelle Projekt- und Elementvorlagen in VS

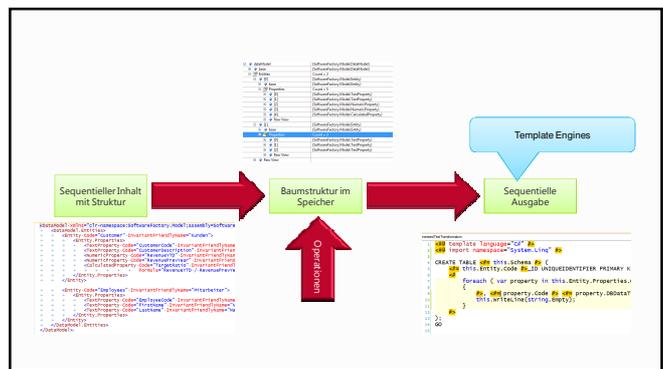
demo

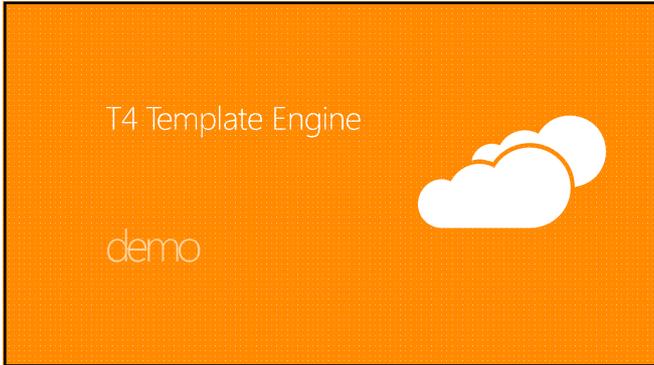


Snippets

- Vorbereitete Codeteile einfügen
- Optional
 - Parameter
 - using Statements
 - Referenzen
- *Code Snippet UI* (Ctrl+K, X)
- *Code Snippet Manager* (Ctrl+K,B)
- Tipp: [Snippet Designer](#) auf Codeplex

Template Engines





Template Engines

Erzeugen beliebige Textausgaben
 Ausgangspunkt ist ein Template mit Platzhaltern (und Code)
 Input ist eine beliebige Objektstruktur im Speicher

Anwendungsbereiche

- Codegenerierung
- Generierung von XML oder HTML
- Compile time vs. Runtime

Zwei Beispiele

- [Microsoft T4](#)
- [ANTLR StringTemplate](#)

ANTLR StringTemplate

Open Source

Implementierungen

- Java
- C# (Full Client und Silverlight)
- Python
- Objective-C

Rudimentäre Debugging-UI verfügbar
[\(String Template Inspector GUI\)](#)

ANTLR StringTemplate Beispiel

```
group DataModelTemplates;
CreateTable(context, entity) ::=
<<
CREATE TABLE [<context.Tenant>]. [<entity.Name>]
(
    <entity.Name>Uuid uniqueIdentifier NOT NULL,
    <entity.Properties:| p | <p.Type.Name>()>; separator=",\n"
)
ALTER TABLE [<context.Tenant>]. [<entity.Name>]
ADD CONSTRAINT DF_<entity.Name>_<entity.Name>Uuid
DEFAULT newid() FOR <entity.Name>Uuid
ALTER TABLE [<context.Tenant>]. [<entity.Name>]
ADD CONSTRAINT PK_<entity.Name> PRIMARY KEY CLUSTERED
( <entity.Name>Uuid )
>>
TextProperty(property) ::= "<property.Name> varchar(50) NULL,
NumericProperty(property) ::= "<property.Name> numeric(18,4) NULL"
```

Deklarativ, keine Schleifen, IFs, etc.

Verschachteltes Template

Microsoft T4 Template Engine

Tief in Visual Studio integriert
Development Time, Compile Time, Runtime
Implementierungen (Runtime) in C# und VB
VS-integrated Editor von [Tangible](#)
[Visual Studio Gallery](#)

T4 Beispiel

```
<#& template language="C#" #>
<#& import namespace="System.Linq" #>
CREATE TABLE <# this.Schema #> (
    <# this.Entity.Code #>_ID UNIQUEIDENTIFIER PRIMARY KEY
    <#
    foreach ( var property in this.Entity.Properties.OfType<PersistedProperty>() )
    {
        <# <# property.Code #> <# property.DbDataType #><#
        this.WriteLine(string.Empty);
    }
    #>
);
GO
```

Sehr Code-lastig! Gefahr, Logik in
Templates zu verpacken.

Vielen Dank für die
Aufmerksamkeit!
